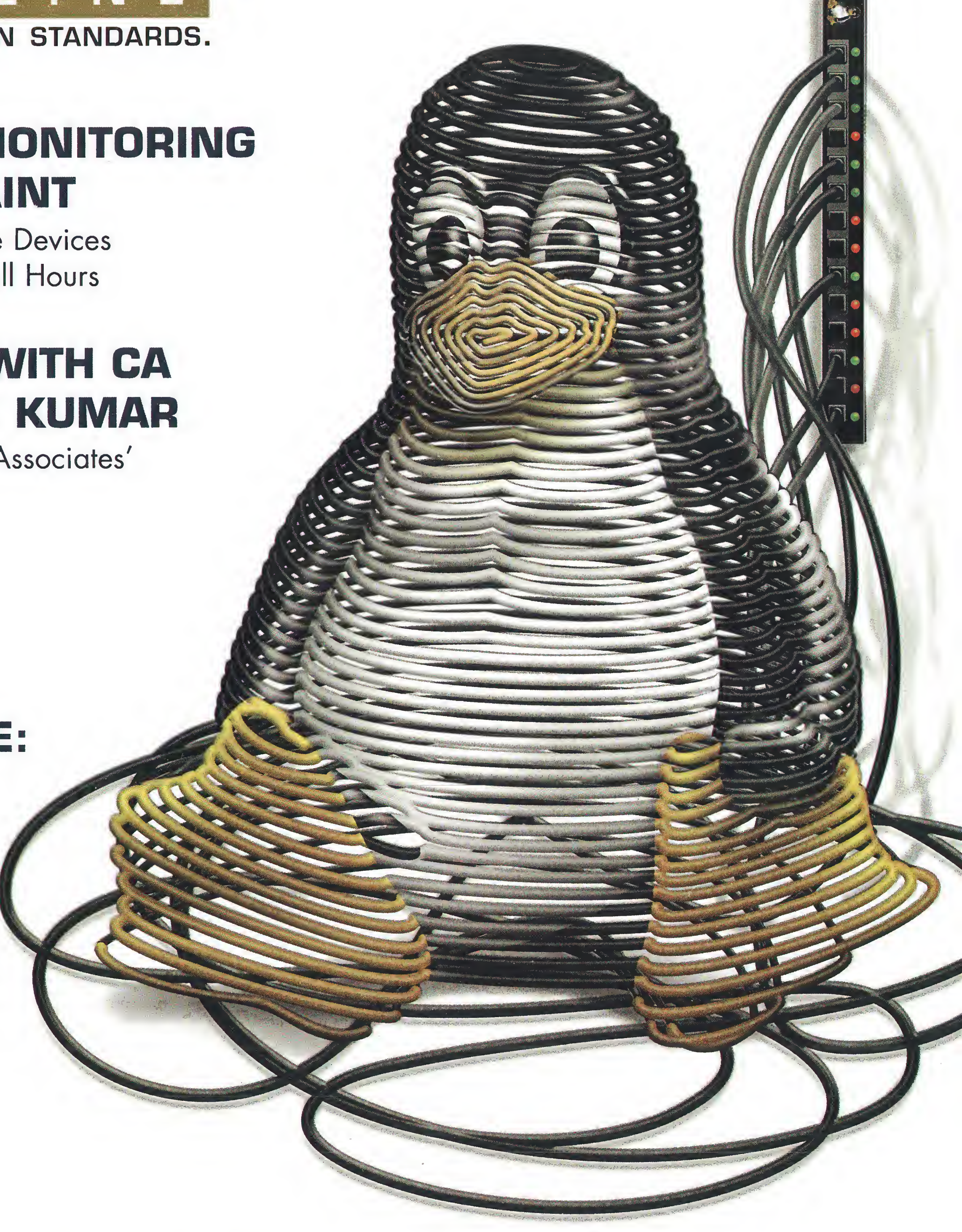
LINUX MAGAZINE

OPEN SOURCE. OPEN STANDARDS.

IPv6: THE FUTURE OF INTERNET PROTOCOL



NETWORK MONITORING WITH NETSAINT

Keep Watch Over the Devices on your Network at all Hours

INTERVIEW WITH CA CEO SANJAY KUMAR

What are Computer Associates' Plans for Linux?

POSTNUKE

A Tool for Creating Dynamic Web Sites

ALSO INSIDE:

- Calculating Spikes in Web Traffic
- Transitioning from bash to zsh
- Managing Mailing Lists with Mailman

MAY 2002

$5.95US $6.95CAN

05>

0 09281 01105 7

WWW.LINUXMAGAZINE.COM

REVIEWED: Borland's Kylix 2 Web Application IDE

Sends other UNIX

"My PowerBook G4 is now running every major UNIX app that we had on our Suns, AlphaStations and SGIs – and running them faster."

–Mark Cohen, Ph.D., Professor, Brain Mapping Center, UCLA

"This new OS has accomplished in a short period of time what others have struggled to do for years: bring a compelling, widely accepted GUI (called Aqua) to UNIX."

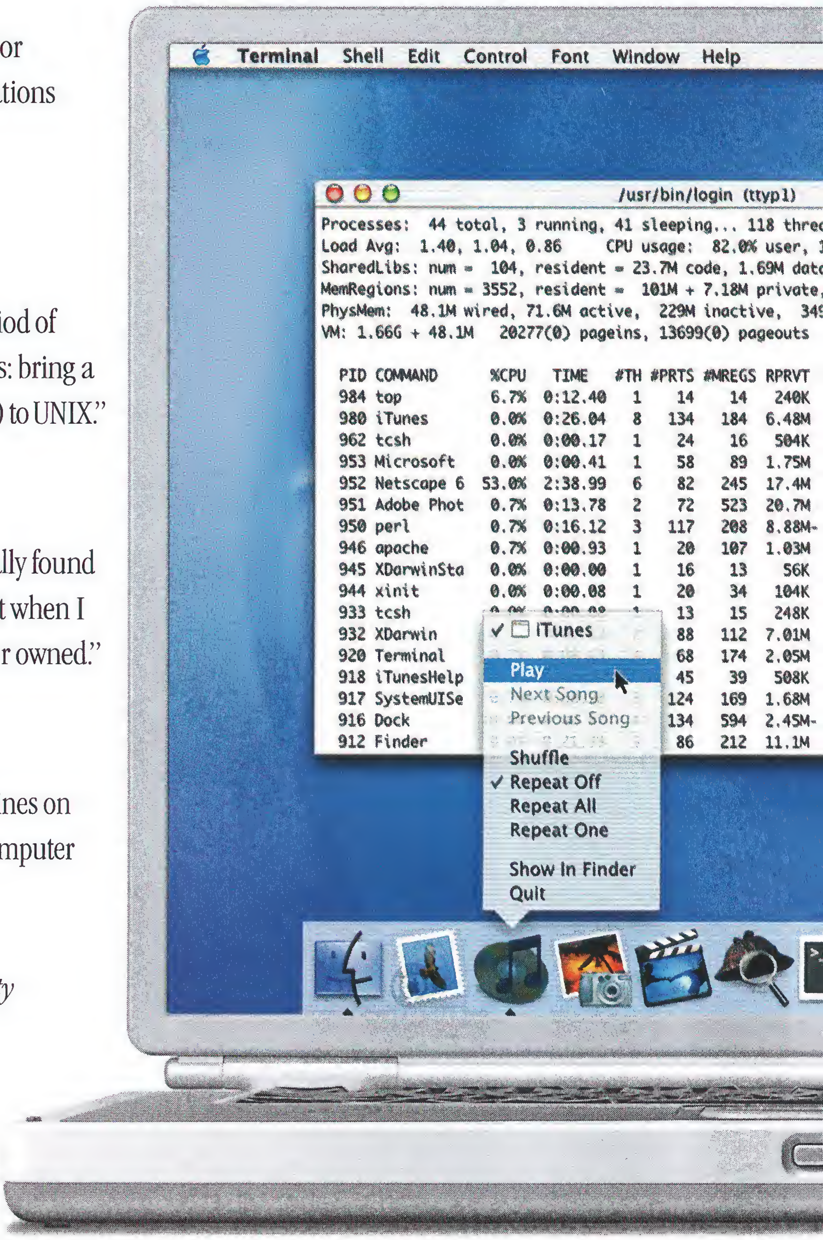
–Tim O'Reilly, O'Reilly Network

"After two-and-a-half years of Linux, I've finally found joy in a UNIX operating system. And I found it when I purchased a Macintosh – the first one I've ever owned."

–John Hummel Jr., The Gamers' Press

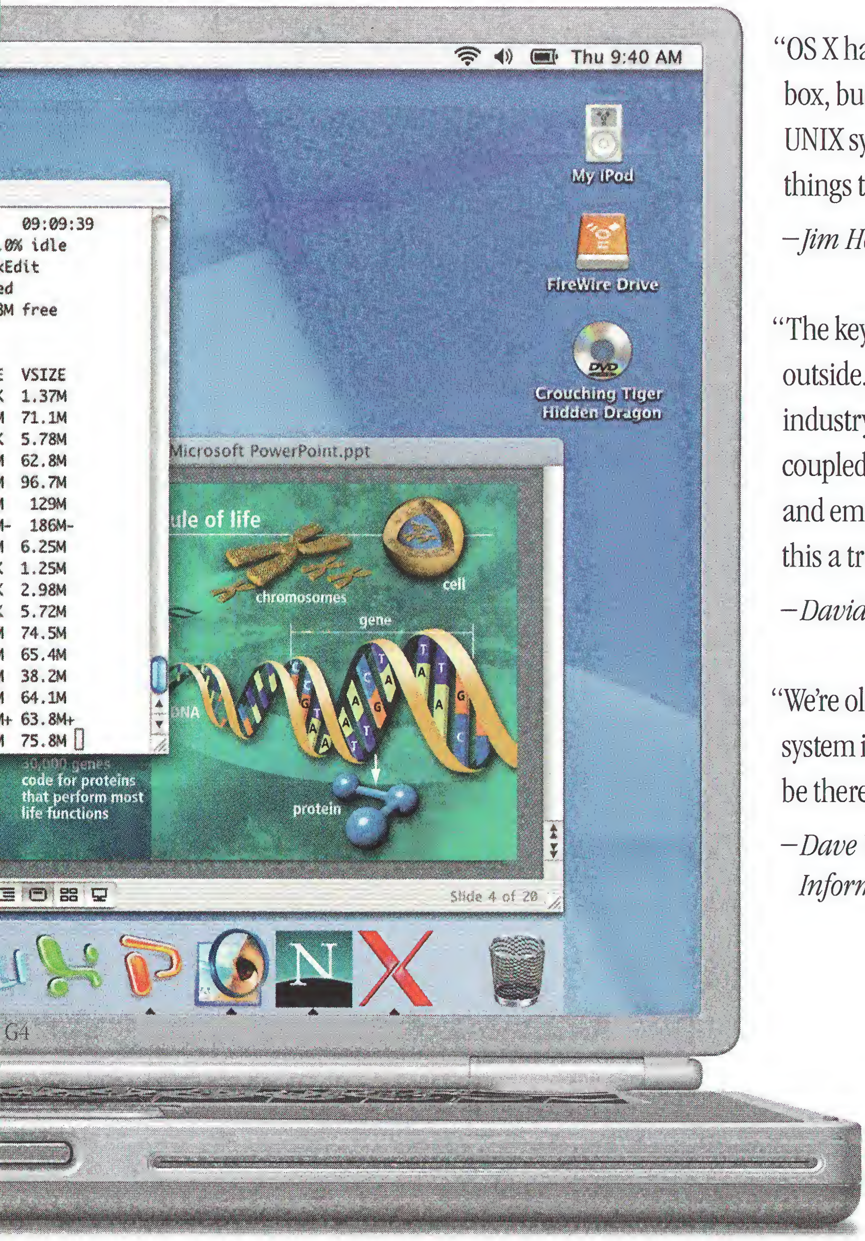
"Until Mac OS X came along, I had three machines on my desk. Now with Mac OS X, I can use one computer for writing, coding, research, everything."

–Dr. Michael Cherry, Associate Professor, Department of Genetics, Stanford University



```
Processes:  44 total, 3 running, 41 sleeping... 118 threa
Load Avg:  1.40, 1.04, 0.86     CPU usage:  82.0% user, 1
SharedLibs: num =  104, resident = 23.7M code, 1.69M data
MemRegions: num = 3552, resident =  101M + 7.18M private,
PhysMem:  48.1M wired, 71.6M active,  229M inactive,  349
VM: 1.66G + 48.1M   20277(0) pageins, 13699(0) pageouts

  PID COMMAND      %CPU   TIME   #TH #PRTS #MREGS RPRVT
  984 top          6.7% 0:12.40   1    14     14  240K
  980 iTunes       0.0% 0:26.04   8   134    184 6.48M
  962 tcsh         0.0% 0:00.17   1    24     16  504K
  953 Microsoft    0.0% 0:00.41   1    58     89 1.75M
  952 Netscape 6  53.0% 2:38.99   6    82    245 17.4M
  951 Adobe Phot   0.7% 0:13.78   2    72    523 20.7M
  950 perl         0.7% 0:16.12   3   117    208 8.88M-
  946 apache       0.7% 0:00.93   1    20    107 1.03M
  945 XDarwinSta   0.0% 0:00.00   1    16     13   56K
  944 xinit        0.0% 0:00.08   1    20     34  104K
  933 tcsh                             13     15  248K
  932 XDarwin                          88    112 7.01M
  920 Terminal                         68    174 2.05M
  918 iTunesHelp                       45     39  508K
  917 SystemUISe                      124    169 1.68M
  916 Dock                            134    594 2.45M-
  912 Finder                           86    212 11.1M
```

boxes to /dev/null.



```
     09:09:39
.0% idle
kEdit
ed
3M free


E  VSIZE
K  1.37M
M  71.1M
K  5.78M
M  62.8M
M  96.7M
M   129M
M-  186M-
M  6.25M
K  1.25M
K  2.98M
K  5.72M
M  74.5M
M  65.4M
M  38.2M
M  64.1M
M+ 63.8M+
M  75.8M
```

"OS X has all the power you expect from a UNIX box, but none of the hassle associated with basic UNIX system administration. You can expect things to work and they do."

–Jim Hourihan, Tweak Films

"The keys to OS X are that it's UNIX inside and Mac outside. This combination of stability, support for industry standards, and wide developer support coupled with Apple's Aqua graphical user interface and emphasis on graphic performance makes this a truly great operating system."

–David Coursey, ZDNet AnchorDesk

"We're old hardcore UNIX hackers, so a BSD-based system is mother's milk. Everything you expect to be there is there, and it works right."

–Dave Weininger, President, Daylight Chemical Information Systems, Inc.

ot included. For more information, call 1-800-MY-APPLE or visit www.apple.com.

Exceptional Clusters, Superior Tech Support

Microway's Reputation is Your Guarantee!

Harness Dual Athlon Power with a Microway® Scalable 25 Node 50 Processor Cluster Using Dual Athlon MP 2000+ and Myrinet Interconnect, Yielding Peak Throughput of 166 Gigaflops.

Microway® Screamer™ Dual Alpha 21264DP 833 MHz, 16 MB Cache in RuggedRack™ Chassis with RRR™ Redundant Power Supply

Athlon, Xeon, and Alpha Clusters

Since 1982, Microway has designed and manufactured state of the art, cost effective solutions for the High Performance Technical Computing community. Our custom Linux clusters and servers are chosen by people at universities, government research labs, biotechs, and financial institutions worldwide. Microway understands that on-time delivery, out-of-the-box reliability, and excellent ongoing technical support are critical to our users. You can have confidence that our solutions employ the latest in processor, connectivity, cluster management, and RAID technologies. Visit microway.com to see our latest products and customer testimonials.

Over 17,000 Satisfied Customers Since 1982 Include:

Corporations and Non Profits
Allied Signal
AT&T
BAE Systems
Carnegie Institute of Washington
Cisco Systems
Cray Research
Disney Studios
EMC Corporation
Exxon
Federal Aviation Administration
Fidelity Investments
Food and Drug Administration
General Electric
General Motors
Grumman Aerospace
Hughes Aircraft
Intel
IBM
Johnson & Johnson
Lion Bioscience
Lockheed Martin
Merrill Lynch
Millennium Pharmaceuticals
Morgan Stanley
Myricom
Nikon
Raytheon Missile Systems
Smithsonian Institution
United Nations
World Bank

Research Laboratories
Ames Laboratory
Argonne
Berkeley Lab
Brookhaven
Lawrence Livermore
Los Alamos
MBL - Woods Hole
Jet Propulsion Lab
NASA
NIH
NIST
NOAA
NSF
Naval Air Warfare Center
Naval Postgraduate School
Naval Research Lab
Naval Surface Warfare Center
Naval Underseas Warfare Center
Oakridge National Laboratory
Sandia National Laboratory
SETI Project
US Air Force
US Army
US Army Corps of Engineers
US Dept. of Defense
US Geological Survey

Universities
Arizona State
Brown
BU
Cal State
Carnegie Mellon
Case Western Reserve
Colorado School of Mines
Colorado State
Columbia
Cornell
Dartmouth
Duke
Duquesne
Harvard
Johns Hopkins
McGill
Kent State
Miami
Michigan State
New Mexico
Northwestern
Princeton
Purdue
Rockefeller
Rutgers
South Carolina
Stanford
SUNY
Syracuse
Tufts
Vanderbilt
Washington (St. Louis)
Yale

Technical Universities
Cal Tech
Georgia Tech
MIT
New Jersey Tech
New Mexico Tech
Rensselaer (RPI)
Texas Tech
Worcester Poly Tech

University of
Alaska
Arkansas
California (Berkeley, UCLA, UCSD, UCSF)
Chicago
Connecticut
Colorado
Florida
Hawaii
Illinois
Kentucky
Maine
Maryland
Massachusetts
Mississippi
Missouri
New York
North Carolina
Pennsylvania
Texas
Utah
Virginia
Vermont
Washington
Wisconsin

Expert Integration ~ Cost Effective Solutions

"Our committee chose Microway over other vendors because of their reputation in the HPC community for delivering high quality, fully integrated clusters. Our decision was based also on their extensive technical expertise with Myrinet, demonstrated financial stability and ability to quickly deliver the 210 node cluster we required."

— Jake Ward, Director, Department of Industrial Cooperation, University of Maine at Orono

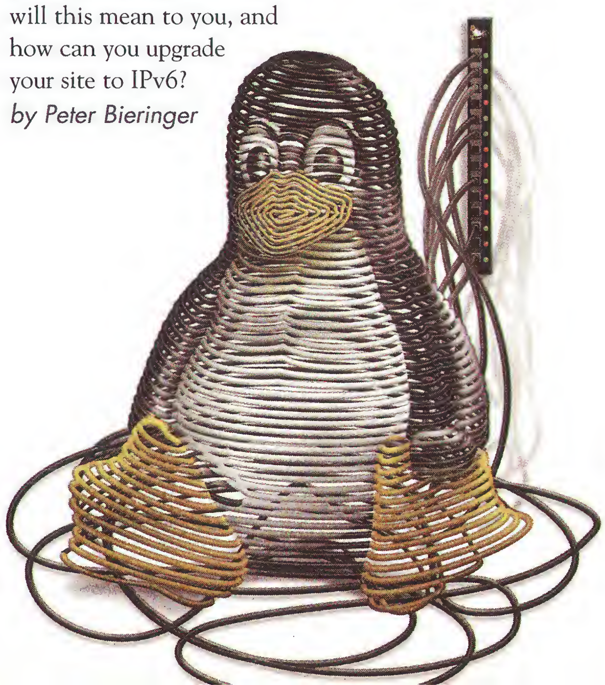
Microway®
Technology you can count on℠

Call us: 508-746-7341
For a technical salesperson who speaks your language
Visit us: www.microway.com

Contents

MAY 2002

FEATURES



DEPARTMENTS

PUBLISHER'S STATEMENT

New and Improved

Welcome to the all-different, all-exciting, all-new version of Linux Magazine! No, seriously, it's impossible to pick up this issue and not notice that it's undergone some serious renovation. More specifically, we've given the magazine a complete make-over, and what you hold in your hands is the result of a re-design that we've been working on for several months now.

Back in September 2001, the editorial staff decided that the Linux market was continuing to move forward in new and different directions. We felt that if we wanted LM to stay current and relevant, we needed to make a few changes to insure that we would continue to best serve our readers as we track Linux's growth and evolution.

Some of the changes we decided to make first appeared in the January 2002 issue, where we announced a few modifications to our lineup of monthly columns. But we felt that we needed to do more than just keep our editorial content up-to-date. We also needed to insure that the "look and feel" of LM reflected the growing maturity and mainstream adoption of Linux itself, which meant completely overhauling the magazine's design.

The re-design project fell into the hands of our Art Director, Christina Empedocles, and the new format you are now looking at is her work. All of us here at LM think she did an incredible job blending the form of LM's design and the function of LM's editorial into a spectacular new package.

Meanwhile, the magazine had one other element that the editors felt could use an upgrade, and that was our tagline "The Chronicle of the Revolution". In our opinion, Linux isn't really a revolution anymore. While it's still a very unique animal, and is part of an incredibly exciting and dynamic movement, it's hard to ignore the fact that Linux has become as mainstream as Unix or Windows. Hardly a day goes by when there isn't some mention of Linux in the New York Times or the Wall Street Journal. And Linux is regularly stacked up against its mainstream competitors when tech publications conduct an evaluation or when an IT shop asks a vendor to bid on a new system.

So we needed a new tagline that more accurately reflected both where Linux is today, and where LM's editorial is focused. We decided upon — "Open Source. Open Standards." — because first, in our thinking, Linux is in many ways a proxy for the entire Open Source movement. At the very least, it's certainly the most visible Open Source project out there. Our second reason for changing the tagline was that our magazine has always been about much more than just Linux. We've always focused on Linux in the context of all the other systems and projects that surround and run on it, and the open standards upon which many of those projects are built. So the new tagline seems to fit our editorial position more accurately than the old one did.

Whew! That's a lot of change for one issue. But then again, that's exactly the point, isn't it? What makes Linux and Open Source so fascinating is the fact that the ecosystem is in a constant state of rapid change, and you can be sure that we'll always do our best to keep up with that. Please always feel free to send us your thoughts at *editors@linux-mag.com*. We look forward to hearing what you've got to say about the "New and Improved" Linux Magazine.

See you next month,

[illegible]

Adam M. Goodman
Editor & Publisher

LINUX MAGAZINE
OPEN SOURCE. OPEN STANDARDS.

Editor & Publisher
Adam M. Goodman

Executive Editors
J.C. Freed
Jeff Okamoto
Martin Streicher

Managing Editor
Christopher Somerville

•

Senior Editors
Jeremy Zawodny
Matt Welsh

Editor at Large
Robert McMillan

Contributing Editors
John Beppu, Joe "Zonker" Brockmeier, Hans Bergsten, Benjamin Chelf, Quentin Cregan, Æleen Frisch, Forrest Hoffman, Jason Perlow, Randal L. Schwartz, Jeremy Zawodny, Steven J. Vaughan-Nichols

Legal Editor
Steven D. Lavine

•

Art Director
Christina Empedocles

Production Manager
Charles MacNulty

•

Circulation Director
Joan Grubb Braunstein

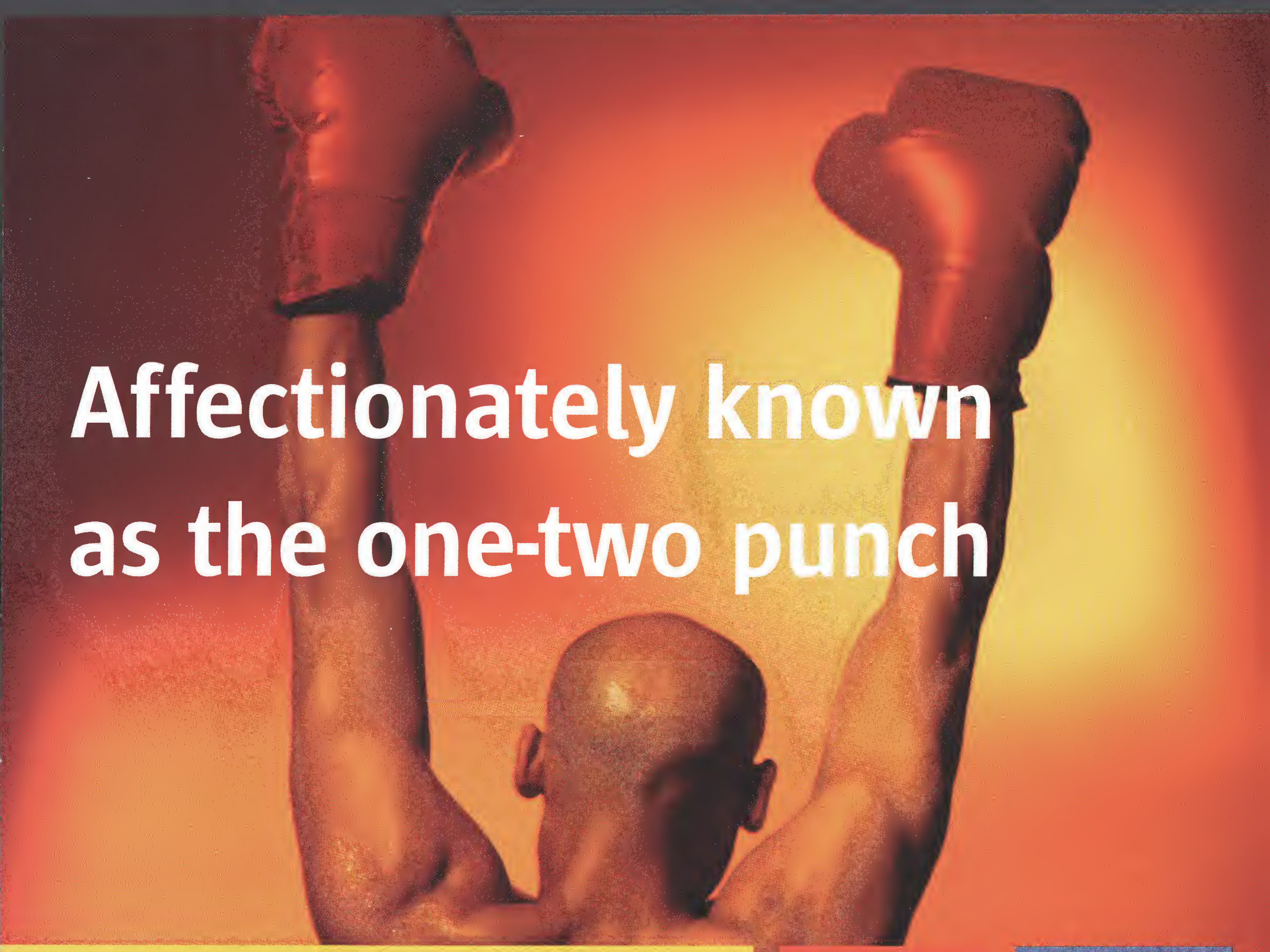
Newsstand Distribution
Irwin E. Billman, Ralph J. Perricelli

•

Web Master
Jon Udell

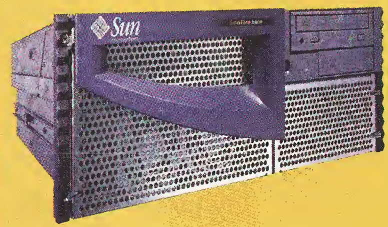
Accounting Services
Jerome Molejona, Justin Stuhaan

BPA Membership Applied For July 2000



The Sun Fire™ 280R and BEA WebLogic Enterprise Platform™. High performance and scalability that will KO the competition.

"How?" You ask. With an open and flexible framework delivered by the BEA WebLogic Enterprise Platform and Solaris™ Operating Environment—the cross-industry choice of Fortune 500 companies. What's more, independent research shows that a Sun/BEA solution costs 2.4 times less than an IBM solution, lowering your total cost of ownership*. Sun and BEA. How's that for a winning combination?



Sun Fire 280R

Looking for a scalable, highly available, rackable server that doesn't compromise on system performance? The Sun Fire 280R server is the smart choice. Based on the latest UltraSPARC III technology, the Sun Fire 280R is an enterprise-class, rack-optimized, dual-processor server, offering exceptional performance and scalability. The Sun Fire 280R addresses customer needs for network-intensive, compute-intensive, or mission-critical applications.

bea®

BEA WebLogic Enterprise Platform

Increase productivity, leverage current and future assets, and lower the cost of your enterprise IT organization with the *only* unified, simplified, and extensible application infrastructure available today. Used by 12,500 customers worldwide, BEA WebLogic Enterprise Platform is the #1 choice for building, integrating, managing, and extending your enterprise applications.

Order Now

Get the free Sun/BEA solution starter kit and receive a 15% education discount when you order by June 30. Contact us for a free TCO study and for more information.

Call today:

1.800.326.6496

Or visit:

www.sun.com/servers/entry/beapromo2

Sun microsystems®

*According to Crimson Consulting Group, BEA TCO Case Study: Large Scale CRM System, February 2002. Results obtained using Sun Enterprise™ 450 server and BEA WebLogic Enterprise Platform.

LETTERS

March 2002

More Highlighting with VIM

I read with great interest John Beppu's March 2002 *Power Tools* column, "Combining VIM's Syntax Definitions."

I use VIM every day to develop Java code. Being able to define syntax highlighting is a great feature, and reading this article prompted me to dig in and attempt to add a simple feature that I've always wanted: highlighting nested parentheses with different colors for different levels.

So I opened Claudio Fleiner's java.vim file and, with a little help from the syntax file tutorial, modified the lines defining parenthesis highlighting to the following:

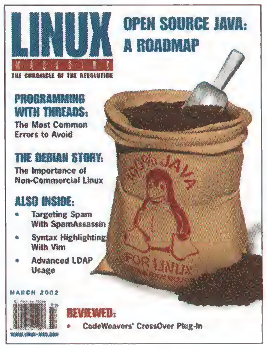
```
syn region javaParen0
   transparent
   matchgroup=javaComment
   start="(" end=")"
contains=javaParen2,@javaTop
syn region javaParen1
   transparent
   matchgroup=javaOperator
   start="(" skip="([^)]*)"
   end=")"
contains=javaParen0,@javaTop
syn region javaParen2
   transparent
   matchgroup=javaBraces
   start="("
   skip="([^)]*([^)]*).*)"
   end=")"
contains=javaParen1,@javaTop
syn match javaParenError ")"
JavaHiLink javaParenError
   javaError
```

This defines three different levels of parentheses, each highlighted with a different color. The colors and number of levels used were arbitrarily chosen. If more than three levels of nested parentheses are used, the colors cycle.

I'd like to thank *Linux Magazine* for providing such interesting material. Great work! I appreciate it very much.

Fred Daoud
Montreal, Canada

Commercially-Minded Linux


LINUX MAGAZINE
THE CHRONICLE OF THE REVOLUTION
OPEN SOURCE JAVA: A ROADMAP
PROGRAMMING WITH THREADS: The Most Common Errors to Avoid
THE DEBIAN STORY: The Importance of Non-Commercial Linux
ALSO INSIDE:
Targeting Spam With SpamAssassin
Syntax Highlighting With Vim
Advanced LDAP Usage
MARCH 2002
REVIEWED: CodeWeavers' CrossOver Plug-In
100% JAVA
FOR LINUX

I've just read Lou Grinzo's *In the Trenches* column, "Communicate With the Masses," on your Web site.

I'm new to Linux (I installed SuSE 7.3 as my first Linux), but not new to the battles of competing computing systems. I remember the emotion and irrationality involved in the battles between 8-bit home computer formats, and later between the Amiga, Atari ST and the developing PC platform. I remember the dread I felt as a teenager when I realized that the Amiga was doomed and that the IBM PC compatible was going to be the dominant platform.

On dipping my toes into the Linux world, I am concerned that the same people who ten years ago were trying to convince me of the divinity of Acorn's moribund Archimedes are now a part of the Linux community. Installing SuSE was certainly easier and quicker than any Windows installation I've made. Linux seems ready to be a mainstream desktop OS to me. However, it suffers from certain sections of the Linux community.

I know a lot of hardcore techies. They've all been great people and some are good friends. However, they mostly share an initial defensiveness that borders on quasi-religious fervor. When this is concentrated into a community, such as that surrounding Linux, situations like the response to your Microsoft ad arise. The fate of the world isn't tied up in one computing platform or another. Innovation will always spring up where huge corporations have stagnated. However, if some members of the Linux community don't realize this, then we are going to appear to ordinary people as a bunch of weirdos.

I think the group development of Linux needs to be extended beyond programming issues and that the community should leave the branding and public voice of Linux to marketing communications experts.

I've also noticed that many Linux Web sites, are often based around the PHP Nuke system. Until there are user-friendly, Linux-based Web sites that make it easy to get Linux info, then I don't see how the system can penetrate the mainstream.

I'm already beginning to love Linux. However, I'm concerned that the community that is so important to Linux may be its downfall in the mainstream market. I think the Linux community needs to be more commercially minded when it comes to its public face.

Matthew Revell,
Marketing Manager
Inco Software Solutions

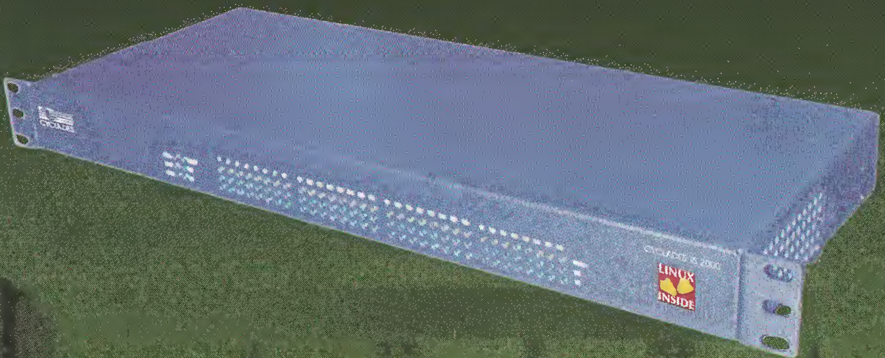
Back Issues Online

Thanks for maintaining such a wonderful Web site. I don't have a subscription to your magazine but I do buy it at my local Barnes and Noble every month. I just started reading your magazine about six issues ago and it is really nice to have access to the past issues online.

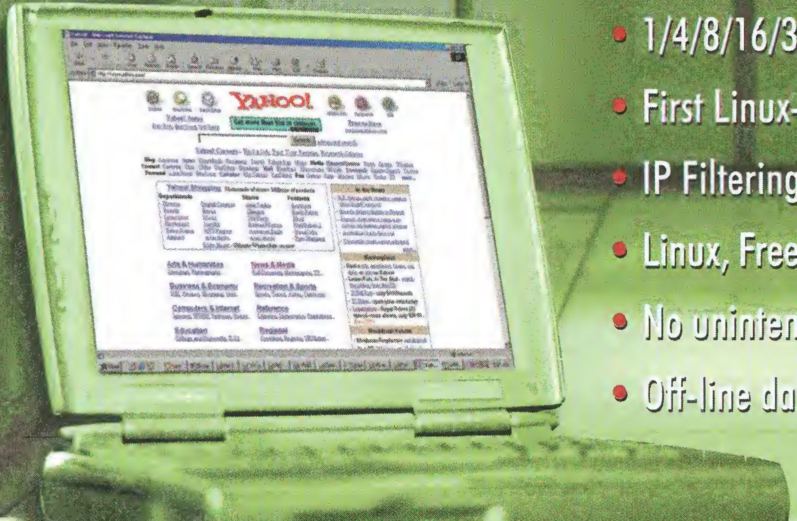
Timothy Carr
Carbondale, IL

Linux Magazine *welcomes feedback and comments. Letters will be edited for clarity and space; please be sure to include your name and location. Send letters to editors@linux-mag.com.*

Guess what Yahoo!® uses to manage their servers?



"The Cyclades-TS Series of Console Access Servers provides the highest port density and security at a very competitive price. By using Linux as the embedded OS, it offers the flexibility required to manage our dynamic environment. The Cyclades-TS is a key element to help us keep our servers up and running." - ***Pete Kumler,*** *Manager of Site Operations, Yahoo! Inc.*



Cyclades-TS Series

Console Access Server

- 1/4/8/16/32/48 RS-232 ports on 1U of rack space
- First Linux-based Terminal Server on the market
- IP Filtering, RADIUS, and Secure Shell (SSHv2)
- Linux, FreeBSD, Sun, HP, and IBM compatible
- No unintentional breaks (Sun)
- Off-line data buffering

SOLARIS READY LINUX INSIDE

Request for your FREE CAS booklet at www.cyclades.com

1999 2000 2001 1995 1996 1997 1998 READERS' CHOICE

THE LEADER IN LINUX CONNECTIVITY

www.cyclades.com

1-888-CYCLADES 1-888-292-5233

sales@cyclades.com

CYCLADES®

Report From The Front

The Latest News From The High-Tech Battlefield

SONY OFFERS LINUX ON PLAYSTATION 2

Popular demand has persuaded Sony Computer Entertainment to sell a version of Linux for its popular PlayStation 2 (PS2) game platform. The kit, Linux for PlayStation 2, includes a DVD-ROM of software (version 1.0 of the kit includes the 2.2.1 kernel), an internal 40GB hard drive, a 10/100 Base-T Ethernet network adapter, a USB keyboard, a USB mouse, and a VGA adapter. The kit allows programmers and hobbyists to develop games and run Linux applications on the PS2 console.

The kit will be available in the United States in June and will cost $199. You must have an 8MB PS2 memory card and a VESA monitor that supports "sync on green" to run Linux on the PS2. http://www.playstation2-linux.com.

MVQ A STREAMING MVP

Finnish developer Oplayo has announced Oplayo Toolkit, an audio and video compression solution based on Motion Vector Quantization (MVQ). According to the company, output from the Oplayo Toolkit is 10 times faster than MPEG-4. Moreover, decoding MVQ requires little processing power, does not require a plug-in, and playback starts instantaneously. A tiny Java applet (20kb) that downloads with the video or audio file performs all the computation required.

MVQ streams are ideal for mobile devices such as pocket PCs and General Packet Radio Service (GPRS) mobile phones. An MVQ stream requires only 14kbps of bandwidth; typical GPRS network bandwidth is approximately 22 kbps. Oplayo's demands on content providers are also slight: streaming servers and dedicated caches are not required. However, if a provider has streaming infrastructure, users will benefit from optimized response times.

MVQ was originally developed by the Finnish Technical Research Center. The MVQ algorithm compresses AVI, WAV, MJPEG, MPEG2, QuickTime, and MP3 files. http://www.oplayo.com.

GIVE ME LIBERTY OR GIVE ME PASSPORT!

IBM is not going to choose between Sun Microsystem's Liberty and Microsoft's Passport authentication technologies. IBM said it won't choose because the company's products have to interoperate with the products of all major vendors.

Liberty and Passport share a common goal: provide a standard, secure method, using a password or other authentication technology, for a computer user to identify himself to all sites on the Internet. Widespread adoption of Liberty or Passport would simplify use of the Web. While Microsoft Passport is in active use, Liberty is still in its formative stages. http://www.projectliberty.org, http://www.passport.com.

LINUX 2.5 KERNEL GETS NEW FEATURES

The Linux 2.5 kernel has added several features, including a new sound driver system and improved handling of real-time programming.

The first feature, the Advanced Linux Sound Architecture (ALSA), provides a professional-quality system for recording, playback, and MIDI sequencing, and supports advanced features like full duplex playback/recording (if the hardware allows) and hardware mixing of multiple audio streams. Developed under the GPL, all drivers included with ALSA are open source.

The second feature adds pre-emption to the Linux kernel. Previously available as a patch, but now officially included in the kernel, pre-emption is expected to dramatically improve Linux's responsiveness. Without pre-emption, code running in the kernel runs to completion, causing latency. Delays as long as hundreds of milliseconds make certain kinds of applications — audio, video, embedded and real-time systems — impractical.

Other new features of the 2.5 development kernel series include boosting scalability beyond the 2-terabyte device limit, and asynchronous I/O, intended to speed up enterprise database performance. http://www.kernel.org.

NEW DISTRO ON THE BLOCK

There's a new distribution of Linux available, and it hails from a company in Redmond, Wash. (No, not THAT Redmond company!) Desktop/LX, from Redmond-based Lycoris, is a new distribution of Linux designed for ease of use and aimed directly at consumers. Desktop/LX looks and feels a lot like Microsoft Windows XP, and it includes personal productivity applications for email, a word processor, a spreadsheet, an Internet browser, and more.

Installation is a snap. You can choose either the basic installation or dual-boot (Windows and Linux), or you can use the expert mode to customize the system to your needs. Installing additional software is a simple point-and-click operation. Desktop/LX includes a seamless integration of WINE (WINdows Emulator) and KDE, and the Desktop/LX desktop environment is attractive and effective. Accessing shared files is done especially well. The Desktop/LX Network Browser mimics Microsoft's Network Neighborhood; just click on the icons to navigate your local network or the Internet.

Desktop/LX costs $29.95. An expanded distribution that includes source code and developer tools is just $10 more. The company offers 60 days of email support. Lycoris also sells Hewlett-Packard desktop PCs and IBM ThinkPad laptops preloaded with Desktop/LX Linux. http://www.lycoris.com.

SUN PERCOLATES NEW JAVA RELEASE

Sun Microsystems has released a new version of its Java 2 Platform, Standard Edition (J2SE). J2SE Version 1.4, available since March, adds many significant features to the core of the Java platform and greatly enhances the performance of important subsystems like I/O. The latest release has over thirty new features or significant enhancements.

Java programmers will be especially pleased with J2SE 1.4. The core platform now includes assertions, logging, and chained exceptions. Assertions allow a developer to validate the state of a program and are a powerful tool for improving code quality. The Java Logging API produces reports that can help developers and system administrators find security failures, performance bottlenecks, and bugs. The new chained exception facility provides a common API to record the fact that one exception caused another. Prior to 1.4, a Java developer would typically create his own implementation of these facilities in each project; now all Java programmers can share a standard and common API .

Other highlights of the J2SE 1.4 release:

- The Java API for XML has been officially added to the Java 2 platform, and provides pre-built code for parsing and transforming XML documents.
- Support for IPv6 in TCP- and UDP-based applications. (See the IPv6 article in this issue for more on the new IP addressing scheme.)
- The new I/O (NIO) API provides PERL-style regular expressions, a file interface that permits locks and memory mapping, and non-blocking I/O.

According to Sun, existing Java source code and binaries are upwards-compatible with J2SE 1.4 (with only a few exceptions). You can download the Java J2SE 1.4 SDK, Java Runtime Environment (JRE), and a pre-release of the Forte for Java 4.0 IDE for free from the Java Web site. http://www.java.sun.com.

SOLARIS ADDS GNOME 2.0 DESKTOP

Sun Microsystems, Ximian, and Wipro are jointly developing a modern desktop for Sun's Solaris based on open source GNOME 2.0. With its next major release of Solaris, Sun has promised to provide both its existing Common Desktop Environment and the new GNOME 2.0 desktop. Users will be able to choose which environment they want to use..

Sun is drawing on Ximian's and Wipro's substantial expertise. Ximian has a history of contributing to GNOME and will extend the work of the GNOME project to create a desktop for Solaris users. Wipro will coordinate the overall effort, provide resources to test the product, and will also integrate the existing Solaris CDE with GNOME. Sun has made several contributions to GNOME 2.0, including accessibility features for the handicapped and language support.

By adding features of GNOME 2.0, Sun will provide Solaris users with a modern desktop and a suite of productivity applications. GNOME 2.0 for Solaris will run on Solaris 8 and higher, and is expected to be available for downloading by October. http://www.sun.com, http://www.ximian.com, http://www.wipro.com.

Kylix 2 Makes Web Application Development a Snap

By Bill von Hagen

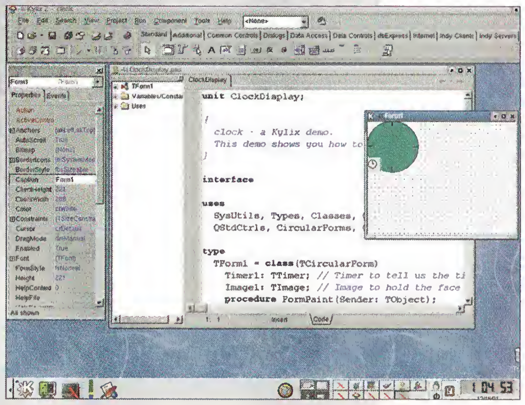
BORLAND KYLIX 2

www.borland.com

Kylix 2 Enterprise:	$1999
Kylix 2 Professional:	$299
Kylix 2 Open Edition:	Free

Rating: 🐧🐧🐧🐧

Pros:
- ➤ Powerful, easy-to-use interface
- ➤ Easy creation of SOAP and XML clients and servers
- ➤ Easy integration with most databases
- ➤ Online help in Bristol Technologies HyperHelp Format
- ➤ Source code for sample applications included



```
unit ClockDisplay;

{
  clock - a Kylix demo.
  This demo shows you how to
}

interface

uses
  SysUtils, Types, Classes,
  QStdCtrls, CircularForms,

type
  TForm1 = class(TCircularForm)
    Timer1: TTimer; // Timer to tell us the ti
    Image1: TImage; // Image to hold the face
    procedure FormPaint(Sender: TObject);
```

Cons:
- ➤ Uses Object Pascal for application development
- ➤ No packaging tool to simplify application deployment
- ➤ No automated integration with Web servers

System Requirements:
- ➤ Pentium 200 MHz processor (Pentium II 400 Mhz recommended, required for Enterprise)
- ➤ 64 MB RAM (128 MB recommended, required for Enterprise)
- ➤ 110 MB disk space (Open), 185 MB (Professional), 225 MB (Enterprise)
- ➤ Distribution: RedHat 7.1 or later, Mandrake 8.0 or later, SuSE 7.2 or later

Attracting developers from other platforms is always going to be a hot-button issue for Linux. Although there are many integrated development environments (IDEs) for Linux, most of them have a steep learning curve.

A possible exception to this "rule" is Borland's new Kylix 2, the Linux sister product of their popular Delphi development environment for Windows. But don't think of Kylix 2 as "Delphi for Linux." Kylix 2 is a development environment with a powerful set of features.

Pick Your Flavor

Kylix 2 is available in three versions: Open, Enterprise, and Professional. The Open edition is a free, downloadable version designed for creating standalone applications that can only be distributed with an open source license. This makes it easy to test-drive Kylix 2, and also shows Borland's sensitivity to the needs and concerns of the Linux market. The Professional edition is designed for commercial use and adds components that ease the development of database-driven and Web-based applications. The Enterprise edition adds a set of components for developing full-blown e-business applications for use over the Web.

Powerful Tools

The heart of Kylix 2 is a toolbar that makes it easy to drag, drop, and customize a wide variety of pre-existing application components, from standard controls and dialogs to powerful data, database, and Internet access components. The toolbar also is how you access many of the new features that differentiate Kylix 2 from both Kylix and Delphi.

The most notable among these is only available in the Enterprise edition. They're three new palettes that highlight the e-business and enterprise application development orientation of Kylix 2. Borland calls these components WebSnap, DataSnap, and BizSnap.

WebSnap provides components for easy development of Web-enabled applications. DataSnap simplifies creating database-oriented applications and middleware, and includes connectors for databases such as DB2, MySQL, Oracle, PostgreSQL, and Borland's own InterBase. Finally, BizSnap makes it easy to create Simple Object Access Protocol (SOAP) and XML interfaces to access almost any kind of new or legacy data.

Drawbacks

With all this power, it's a shame there are two significant issues that could negatively impact your decision to use Kylix 2. The first is that Kylix 2 applications must be written in Object Pascal. Though Object Pascal brings some of the advantages of Pascal to object-oriented programming, it's not very widely used. The other is that Kylix 2 doesn't provide any packaging tools to deploy your application and integrate it with an Apache Web server.

Try it Out

Kylix 2 shows that Linux is truly coming of age as a development platform. The power, productivity enhancements, and cost savings that Kylix 2 can bring to your development organization makes it much more than just a potential alternative to Windows-based development — taking Kylix 2 for a test drive may just be the smartest business decision you'll make this year. **LM**

For Years to Come...

In turbulent and changing times, it's nice to know that some things remain unchanged. Here at ASL, we take pride in the fact that we have been a premier computing partner choice since 1995. Whether accomplishing a research project, setting up your company network, or maximizing your computing performance, ASL will continue to be there. Let us deliver the very latest hardware technology optimized for your computing needs utilizing the most sophisticated open source software solutions—all in a cost-effective package.

We welcome you to benefit from our vast technical knowledge and continuity.
We welcome you to the continuing ASL experience.

workstations • servers • open source

call 1.877.ASL.3535 or visit www.aslab.com

asl
ReDefining Linux Performance

Mailman, the GNU List Manager

http://www.list.org/

For nearly as long as folks have used the Internet for sending and receiving e-mail, mailing list management (MLM) software has been around. If you're looking to set up a mailing list server, Mailman is probably just what you need. It is popular, fast, easy to use, and easy to hack on.

Mailman hasn't been on the scene as long as *Majordomo*, *ezmlm*, or other MLM software, but it has quickly been adopted by some of the most popular Open Source companies and projects. In fact, Mailman is already used by the following companies, just to name a few:

- Apple Computer
- SourceForge and VA Software
- Freshmeat
- Debian
- The XFree86 Project
- The Samba Project
- RedHat Inc.
- Python.org
- Gnome.org
- Exim.org

There are thousands of Mailman installations around the world, and the number is growing every day. Clearly there must be good reasons for its popularity.

Features

So what makes Mailman special? It's not just the fact that it's a real Open Source-style project. It has a feature list that is only matched by commercial (and expensive) MLM software. Specifically:

WEB INTERFACE

Mailman automatically provides a customizable home page for each mailing list. Users can subscribe and unsubscribe themselves as well as changing list preferences.

List and site administrators can use the Web interface for common tasks such as account management, approvals/moderation, list configuration, and so on. But don't be misled by the role of the Web in Mailman. The Web interface isn't a replacement for command-line or e-mail based tools. Most tasks can be performed from the command line using shell scripts or as commands embedded in e-mail messages sent to Mailman (like *majordomo*).

DELIVERY OPTIONS

Each subscriber may select the way mail is delivered to them. Mailman can perform normal (single message) delivery as well as MIME and plain digest formats.

LIST ARCHIVING

By default, Mailman can archive all list traffic and provide a Web front-end to browsing and searching messages. If you prefer to use other archiving tools (such as *MHonArc*), you can set up Mailman to work with them. The Mailman-generated archives can be public or private (accessible only by list members).

BOUNCE DETECTION

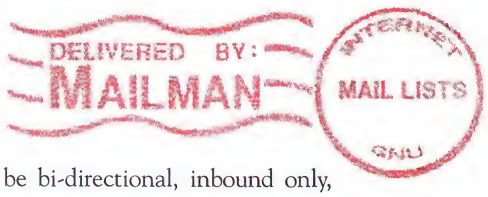
Even without tweaking its settings, you'll find that Mailman does a good job of figuring out when a user is having trouble receiving e-mail. If the problem persists, Mailman will automatically stop delivering to that user.

MULTIPLE ADMINISTRATORS

Lists can have multiple administrators, making it easier to share the work on large or exceptionally busy lists.

NEWSGROUP GATEWAY

One of Mailman's cooler features is its ability to relay list traffic through an NNTP (Usenet News) server. On a per-list basis, you can configure relaying to be bi-directional, inbound only, or outbound only.



There are many other features that make Mailman a competitive MLM. It has many of the same features present in older MLM software, as well as a lot of Web-based functionality that they don't. And unlike other software, the Web front-ends for Mailman weren't added as an afterthought.

Requirements

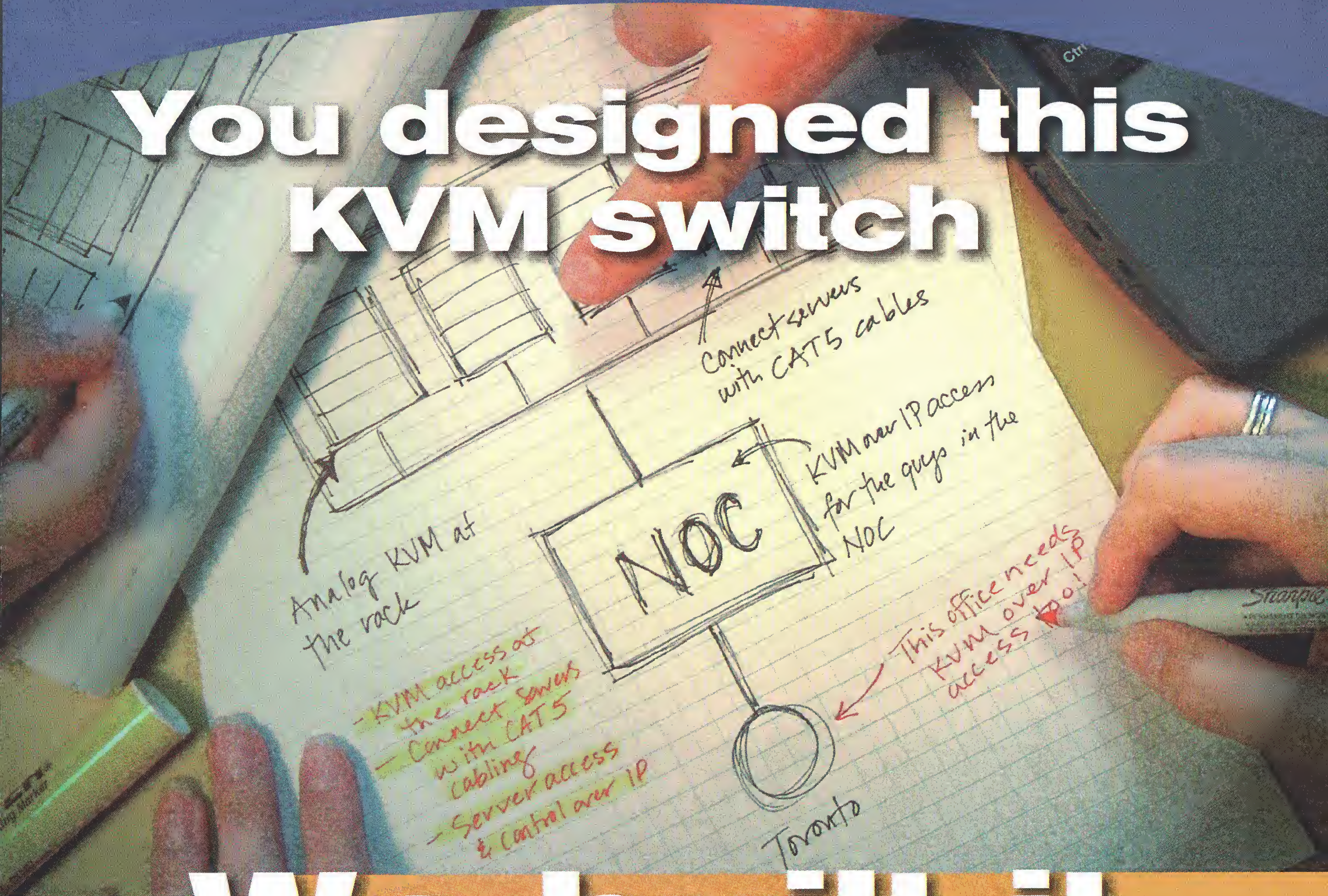
Mailman, unlike some other MLM software, is easy to install. It isn't very picky about the software it'll work with. Of course, you'll need to have a Mail Transfer Agent (MTA) installed. Mailman doesn't care if you use *Sendmail*, *Exim*, *Postfix*, *Qmail*, or anything else. It's not closely tied to the MTA (unlike some mailing list managers), so it's not a problem if you need to move your Mailman installation to another machine or switch MTAs at some later date.

Most of Mailman is written in Python. It will work with older versions of Python (1.5.2, 1.6) as well as the newer 2.x series. Hacking on it is relatively easy.

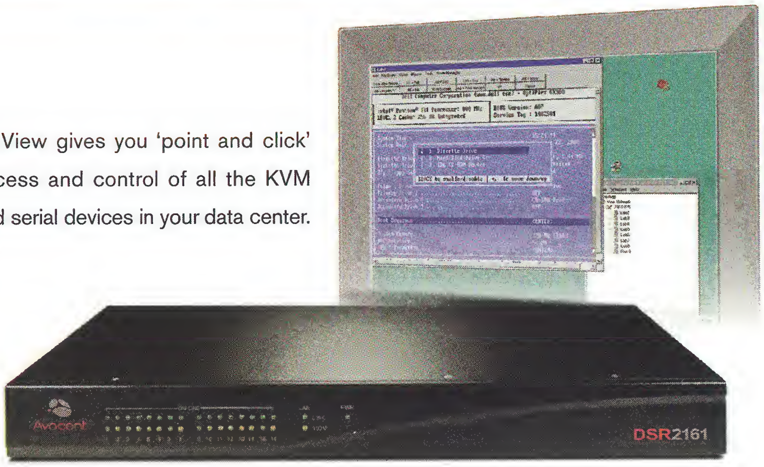
For the Web interface, you'll need a Web server that can run CGI scripts. While you're not required to use any particular server, Apache is recommended.

All in all, Mailman just goes to show that Open Source hackers can deliver powerful and easy to use tools that rival the commercial alternatives.

Have an idea for a project we should feature? Drop a note to potm@linux-mag.com and let us know.


You designed this KVM switch
Connect servers with CAT5 cables
KVM over IP access for the guys in the NOC
NOC
Analog KVM at the rack
This office needs KVM over IP access too!
- KVM access at the rack
- Connect servers with CAT5 cabling
- Server access & control over IP
Toronto

We built it.



DSView gives you 'point and click' access and control of all the KVM and serial devices in your data center.

With the new DSR analog/digital KVM switches, your choices for access and control have never been better.

See our latest designs at www.avocent.com

Avocent™
The Power of Being There™

For a FREE Smart Server Management Guide and more information about our connectivity solutions, visit www.avocent.com or call 1-866-AVOCENT (286-2368).

compare and

"I'm really impressed with the program and the site. This site has the potential to become the "Holy Grail" of Microsoft hosting information resources. Keep up the good work!"

– Darren Andrews
President
InfoTech Inc.

*Membership benefits are contingent upon the successful completion of the application and training requirements of the Windows Web Hoster Program.

Microsoft® Service Providers

understand the technology

The Microsoft® Windows® Web Hoster Program's online community provides ISPs with the software, support, and technical resources necessary to successfully manage their Windows-based shared and dedicated hosting services more efficiently. Learn how to deploy infrastructure, optimize performance, and provide better service and support to your customers.

Windows Web Hoster Program Benefits*:

- A **FREE** copy of Microsoft Windows 2000 Advanced Server.
- A **FREE** copy of the Windows 2000 Server Resource Kit on CD.
- **SPECIAL DELL OFFER – $200 Off** the award-winning Dell PowerApp.web™ 120: a turnkey, dedicated Web server featuring headless deployment and secure, browser-based administration. Offer includes Silver and Direct Line Support.
- Access to the private Windows Web Hoster Program community Web site.
- **FREE** 24-hour, 7-day-a-week access to Microsoft technical support engineers through the Microsoft Direct Access Business Critical Support Program.
- One-on-One Microsoft FrontPage® Server Extension support through the Microsoft Web Presence Provider Program.

To learn more about the Windows Web Hoster Program, visit: www.microsoft.com/isp

Technical Resources and Support Benefits:

- **HelpShare:** Ask a question; set the reward; get fast, accurate answers. Or...answer a question and get paid!
- **Community Chat:** Twice a month you have the opportunity to engage directly with Microsoft MVPs on Web hosting topics.
- **Business Critical Phone Support:** Get expert answers to your problems quickly (Passport registration required).
- **Web Hoster Newsgroups:** Post your Web hosting issue and receive a response from other community members as well as Microsoft Product Support Services.
- **Microsoft Knowledge Base:** Search the Microsoft Product Support Knowledge Base for product and technology solutions, self-help tools, and how-to information.
- **How-To Articles:** In-depth technical information about using Microsoft products and technologies in Web hosting scenarios.
- **Community Bulletin Board :** An interactive forum for peer-to-peer communication by community members.

Microsoft® Windows 2000 Server Family

Microsoft®

www.microsoft.com/isp

ca
Computer Associates

Associating With LINUX

Computer Associates' **Sanjay Kumar** Sees Big Business

By Robert McMillan

So what does the CEO of the world's fourth-largest software company think about an open source phenomenon like Linux? Well, for one thing, that the "open source" side of things is not the interesting part. What has Sanjay Kumar all fired up is the enthusiasm CA's enterprise customers — mainframe users who wouldn't give Linux a second glance two years ago — suddenly have for the platform. And Kumar should know what the mainframers are thinking — they make up about 50 percent of CA's business. That enthusiasm (for Linux, not free software) has caught on at CA, which claims it will have over 50 applications ported to Linux six months from now. Linux Magazine's Adam Goodman, JC Freed, and Robert McMillan caught up with Kumar in New York at the January 2002 LinuxWorld Expo to ask him about Linux in the enterprise, the GPL, and the $64,000 question: whether or not IBM should do its own distribution.

LINUX MAGAZINE: What did you first think of Linux?

SANJAY KUMAR: My first question was, can I make money from this? I mean, that's what this is all about, right? And I had to be convinced. If you want me to say, "Sign up to invest in this thing," somebody has got to convince me that the world is going to support this platform. And it took a few rounds to get there. It was as much an exercise for us, internally, to think about the platform and do some research on it.

I started talking to customers, because I believe in having a lot of independent views on things, and what I found at that time — 1996-1997 — was that high-end customers didn't want anything to do with it. Low-end customers were incredibly excited because of the price — especially if someone was going to support it. And mid-tier customers were willing to buy bundled solutions. In other words: "I want a turnkey solution. I don't care what it runs on, I just want it to work. I'm buying the application."

And so in 1998, when we started doing more work on it, I expected the low end to be there, the midrange to be bundled with applications, and the high end to be non-existent. And I can tell you I was completely wrong. The high end is where the action is today. I mean, you have the core Linux community, right, that's been spread around: either the mid tier, the low end and some of the high end. But in terms of who's willing to spend money, where we thought we would start and where we are today is night and day.

LM: So what happened? What enabled that?

SK: A lot of CIOs were being talked to about Linux by technical

"I was completely wrong. The high end is where the action is today."

people, but the pushback was, "But if it breaks, who do I call? If something doesn't go right, you know, who do I blame? There's no one person behind it." And that's very unnerving to people. There has not been a technology around that has been incredibly popular, that's not had somebody whose feet you can hold to the fire and say, "You know what, if you don't fix that problem, I'm not buying anything from you in the next quarter and you're going to suffer." That's a powerful hammer. People didn't have that [with Linux], and that's a major cultural adjustment for customers, because they want accountability. And when IBM stepped into it last year and said, "This is kosher; we'll stand behind it," I think that brought a lot of people to the edge, to say, "OK I'm willing to really consider this now." That was the big event.

At that point we authorized the cranking up of two development labs for Linux and porting a whole bunch of products. And we're seeing the fruits of that labor today. We placed a bet that IBM was going to get behind it very seriously, and they did.

THE BACK DOOR

LM: IBM's sponsorship of Linux may appear as a double-edged sword to you. On the one hand, they legitimize the platform for your customers, but on the other hand, do you feel that there's a risk that IBM may somehow be overly associated with Linux ...

SK: You mean hijack the platform?

LM: ... If you want to call it that. But also, IBM seems willing to commoditize a lot of lines of business, so that could also present a problem for companies like yours.

SK: We have competed with IBM for 25 years. We started with one product. We sold it and they gave theirs away for free, so the idea of them bundling — while it's always a formidable thing to compete with — doesn't bother me because ultimately the customers are going to pay for value. If you've got differentiation and value, [customers] will buy. We've learned that the hard way over 25 years.

The one thing that's counter to the argument about any one person hijacking the platform is, by definition, the platform's portability. I think what will happen if customers start to deploy on [IBM's] Z platform for Linux and the hardware gets too expensive, people are going to say, "That's fine. You know what? I'm going to invite Dell, Compaq, and HP to come in and bid a nice 16-way cluster server set to just move the platform over." And a number of customers that we're working with today — high-end household names — are using the Z platform to develop and test on, but will deploy on Intel.

LM: Why is that?

SK: Because they believe, from a pricing perspective on hardware, the Intel platform is a commodity and the Z platform is a monopoly. Think about that. There is only one choice for Z.

I can tell you there are some large customers who six months ago were committed to deploying on Z. I had a conversation with one of them, a super-large customer downtown in the city [New York], who told me, "Well, we have changed our mind. We will develop, test, and maybe even deploy round one, but we will never deploy *en masse* on that platform."

LM: Do you think Linux will ever be the dominant OS on the mainframe?

SK: No, it will never dominate. There's too much legacy; there's too much in terms of applications that have been built; there are too many sophisticated mission-critical systems still. MVS is MVS is MVS. It does things that other things just can't do. It's just a fact of life. While there are many examples — Amazon's one — of high-end transaction providers, you know, consumer companies who are running Linux, if you want to know transactionals, you've got to go to a SABRE or a Charles Schwab and sit there when the New York Stock Exchange opens to see real transaction workloads. It's a whole different league.

If not for the mainframe, I don't think some of the companies doing work with us today would have even gone to Linux. And that's its back door into the enterprise. What it's going to do is get them to Linux, and I don't think they will stay on the mainframe forever. I think they will become comfortable with the platform, and [then] they're going to go off somewhere else.

LM: What do you think of the GPL? Do you think it's contributing to the autonomy of the Linux platform?

SK: I think it does, but the platform is still — in commercial terms — too young, too immature to figure out where it's going to go. We're seeing the beginnings of a whole new movement here. I don't think it'll be as big as the Internet, but it could be meaningful in a different way.

People say you can never have a proprietary extension to Linux. That's BS. Of course you can. Nothing prevents IBM from coming out with add-ons to their Z platform in microcode that makes Linux do different things. Sure it's completely permissible; there's nothing wrong with it. So I think it's too early to tell where it will end up.

The more competition there is, to some degree, the more pressure there will be on the vendors to do unique things. To some degree customers want competition in the marketplace, but the more customer competition there is, the more vendors have to differentiate.

OPEN SOURCE PLANS?

LM: Have you released open source code?

SK: Yes, in two instances. They are both extensions to our management products.

LM: What license did you use?

SK: The GPL.

LM: That seems like an area where there's a difference between CA and IBM. IBM's released a lot of Open Source code and they keep moving into these new spaces, with Eclipse for example. In your LinuxWorld keynote, you intimated that more community things would be happening at CA. Can you tell us a bit about them?

SK: We will never open source everything that we're doing on this platform. I'm really clear about that. That's just the way it is. I will also never go to the other extreme and say, "Nothing will be." I think there are a whole bunch of things that make sense. The problem is, you've got to be careful in what you do. I make security software. It runs in a hairy space, right? I would be doing an absolute disservice to turn that stuff loose. My customers would just go nuts. They would absolutely come unglued if I did that. I do believe that there are pieces of that technology that do make sense to be out there — where we can get other people to help us — but we have to re-engineer some of the base technology so I can separate them.

LM: So in what areas do you think it makes sense for CA to open source software?

SK: I think on the management software side, the agents that manage the platform. The network element technology, for example, that's rapidly

"I don't want to be in the distribution business, because I'm platform-agnostic."

changing. That makes sense. There are components of the database products that make sense. I would never turn loose the optimization engine, for example. That I consider truly proprietary. But our Ingres database product was developed in a university community environment. I don't have a major problem, in principle, for it to go back there. But the optimization engine, which we have invested 20 years in building, I have a problem with that [being open sourced]. That's my competitive differentiator. There ought to be a clean way to talk to it, there ought to be published standards, I ought to make those standards open, and I ought to say, "This piece is mine, but all these other things I'm going to turn loose." But I have to get the technology [set up] in a way where I can unbundle and separate them. If not, I create a big mess.

CIOS AND OPEN SOURCE

LM: We've been talking about Linux a lot, but to what extent do you think the phenomenon is really about open source, the development methodology?

SK: This is probably the statement that will most disappoint your readers. I don't think it's got much to do with that whole open source thing right now. The space that we're in, it's not about open source. It's about customers coming to us and saying, "This is a terrific platform. We never realized it could do the things it could do, either from an embedded side, which is completely customizable, or from the really high-end side. There's no other platform, short of going in really haggling, begging and pleading with the vendor, where we can get it to do that." If people came to us and said, "Can you customize Unicenter and make it this big or this big or this big?" I'd go out of business if I did that for every customer. So a big driver today is customers saying, "I didn't realize you could do this." There's this surprise factor. "I didn't realize that it's stable and IBM has embraced it." So to some degree, the purist view is not what's driving it. Ultimately, it's an attempt to save money.

LM: So does the open source nature of Linux interest your customers at all?

SK: At the CIO level, no. For them, it's stability, price performance, efficiency, viability, who else is doing it.

LM: And they couldn't care less that it's open source?

SK: I know it's going to disappoint a lot of people, but no. Two levels down in the organization, it makes a huge difference.

POSITIVE CONSOLIDATION

LM: How many Linux distribution vendors do you think there will be in five years?

SK: Less than today. I think consolidation is good because it will do a couple of things: It will make the ones that survive the first round — and we're still in the first round — stronger. It will send a powerful message to customers that consolidation is permissible in this marketplace and viable. Right away people think, "Customers like bigger things." Not necessarily. Customers also want to know that the guy who can survive is valuable enough for somebody else to absorb. That's critical in this day and age. It was not critical two years ago when the market was all up and flying, but today it's critical. A customer wants to know that XYZ Company that I bought stuff from is viable [and] even if they can't make it, that somebody's going to pick it up, which makes the platform look and sound safer. I think if the big hardware guys become providers, distributors, supporters, that's important. Customers look to them naturally. That's where the platform has come from for as long as we've known computing, with the exception of Microsoft.

LM: So are you saying that you think IBM should do its own distribution?

SK: Well ultimately I think that customers would love for them to.

LM: They don't want to do that, though.

SK: I know. Customers would just love for them to.

LM: And would that bother you?

SK: It wouldn't bother me one bit. I don't want to be in the distribution business, because I'm platform-agnostic. With Linux, you have this — it's not an issue, it's not a quandary — but you have a particular circumstance where you have a community that feels that they own the platform and then a bunch of capitalists have shown up around it. Without the capitalists, the platform ain't going to go anywhere. And without the people under you who have done it, the platform never would have gotten [to where it is]. So you need both to get along, and the question is where is the happy medium. The two halves of the community have to become one. Without one, the other can't survive. People are very hung up about what one's interest is over the other. It really doesn't matter, because as far as I'm concerned, if you don't have both, it just won't work.

Robert McMillan is Editor at Large for Linux Magazine. He can be reached at bob@linux-mag.com

ENTERPRISE NETWORK BACKUP

Save Your Data... Save Your I.T. Budget

IT'S ESSENTIAL to protect what's important to you. And saving your I.T. budget is just as crucial. That's why your number one solution for data backup and restore is Arkeia.

As the leader in Linux backup, Arkeia is reliable. It's also quick. And priced affordably. It's not surprising that over 75,000 users worldwide rely on Arkeia to save their data.

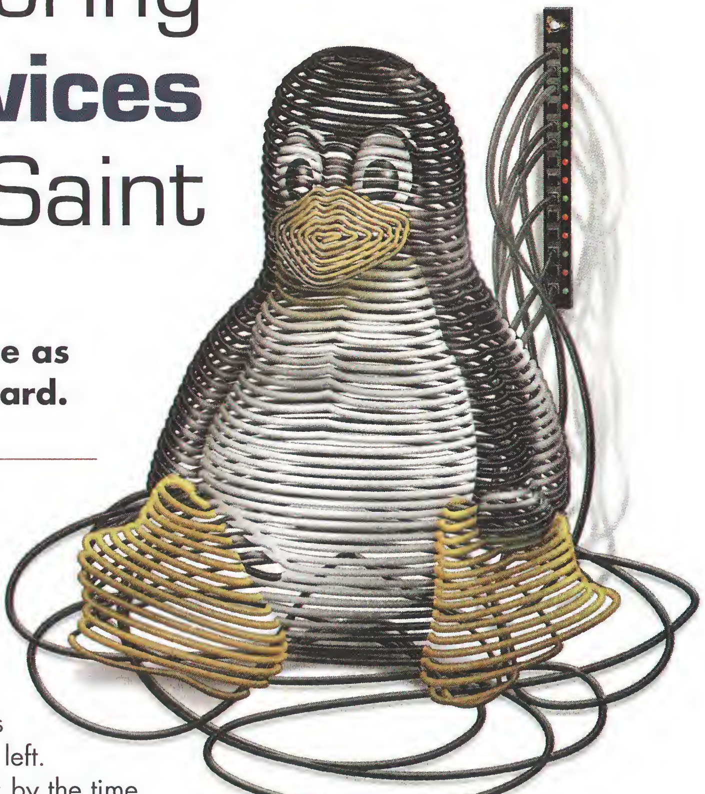
Arkeia's ease of use has been awarded the Maximum Linux Power Award, the C|NET Editor's Choice Award, and the Linux Magazine Editor's Choice award two years in a row.

See how thousands of other corporate customers are saving their budget by protecting their data with the more affordable Arkeia.

Demo Arkeia FREE when you go to www.arkeia.com/demo or call today – 888.333.KNOX.

arkeia® by knox software

Monitoring **Network Services** with NetSaint



Checking Your Devices' Status Can be as Easy as Checking Your Car's Dashboard.

By Æleen Frisch

Like many people, you might not pay as much attention to your car's fuel gauge as you should. You might only notice the gauge when the gas tank is nearly empty. Some new cars, though, have a nice feature: a dashboard indicator light that reads "Low Fuel" when there is about an eighth of a tank left. This strategy is much better than, say, the "Oil" light; by the time *that* light comes on, the car is already in trouble.

Network monitoring has tended to be more like the oil light than the low fuel light. Too often, the first indication of a problem is some failure: a disk partition has filled, a process has crashed, or a critical server has lost its Internet connection. And like a neglected car, a failure in your network inevitably requires an immediate and often difficult, costly, or time-consuming repair.

So, how's *your* network right now?

That's probably a tough question to answer. And no wonder. Even the smallest enterprise is likely to have tens of critical services it depends on: domain name service, sendmail, POP servers, FTP, Web servers, file servers, printer spoolers, database servers, VPN servers, and a collection of hardware tuned for specialized tasks. While it is possible to monitor your computing resources and services — just as you can always check your car's oil level — doing so is something that can slip through the cracks in a busy schedule. Keeping track of all that software (and hardware) becomes an increasingly daunting task as the number of computers, devices, and services you're responsible for increases.

Standard Linux distributions provide little relief. The small number of monitoring utilities that are included — commands like *uptime* (which measures CPU usage), *vmstat* (which measures memory usage), and *ping*, *traceroute* and *netstat* (the latter three measure network connectivity) — are generally limited to examining a single machine and its own network connectivity. Fortunately, a class of utilities called network monitors offer a solution.

Network monitors actively track and report the status of groups of computers and other network devices: printers, routers, and so on. Most network monitors will keep you well-informed with a Web interface that is updated frequently. In some cases, you can even track performance data. (Other packages, that monitor traffic on your network, are also referred to as network monitors, but aren't covered in this article.)

Most importantly, network monitors keep a vigilant watch over each resource, looking for problems: a system or service that's become unusable (e.g., basic connectivity tests fail) or the value of some metric has moved outside its acceptable range (e.g., the load average on a computer system rises above some preset level, indicating that CPU resources are becoming scarce). In these situations and others, the network monitor will notify you (via e-mail or with some kind of visual indication) about the potential problem, allowing you to intervene before the situation becomes critical. The most sophisticated programs can also begin fixing some problems as soon as they are detected.

In this article we'll look at one popular monitoring package called NetSaint (http://netsaint.org), written by Ethan Galstad. NetSaint is designed to run under Linux, but should work on other UNIX variants, too. NetSaint can run as a normal process or as a daemon, and network status can be conveniently viewed through any Web browser. You can easily extend the features of NetSaint and can customize its reports to match the topology of your network.

THE (NET) SAINT

NetSaint is a full-featured network monitoring package that provides information about system and service status across an entire network. If run as a daemon, NetSaint runs periodically, probes devices and other daemons, gathers data, and then summarizes the current state of your network in a number of customizable system status reports.

NetSaint itself is simply a control program; all of the actual system and service monitoring is performed using external plug-ins. Hence, it's easy to extend NetSaint's capabilities by adding your own plug-ins. NetSaint can also be configured to send alerts and perform other actions when problems are detected (see "Network Monitoring Solutions" for other tools that you can use).

Installing NetSaint is straightforward. Choose a machine to host NetSaint and download the free software. RPMs are available for many popular Linux distributions and recent SuSE Linux distributions include NetSaint (although it installs the package in a non-standard location). Like most network monitoring packages, NetSaint has several prerequisites (including MySQL and a Web server); be sure to read the "What's New" section of the NetSaint documentation before you install anything.

These are the most important NetSaint components:

- The *netsaint* daemon, which continually collects data, updates displays, and generates and handles alerts. The daemon is usually started at boot time by placing a link to the *netsaint* script in */etc/init.d/*.

- Plug-in programs, which perform the actual device and resource probing. You can create your own plug-ins or re-use those written by others. In any case, you'll need NetSaint plug-ins before you can actually monitor anything.

- Configuration files, which define the devices and services you want to monitor.

- CGI programs, which support Web access to the displays.

Figure One displays NetSaint's Notifications screen. It provides summary information about the current state of everything being monitored. In this case, we're monitoring five hosts, of which four have some kind of problem. The screen shows the host, service, and state of the service. Some services have many entries — more than one person can be notified in case of failures. The display shows an abnormally high number of failures to make the discussion more interesting.

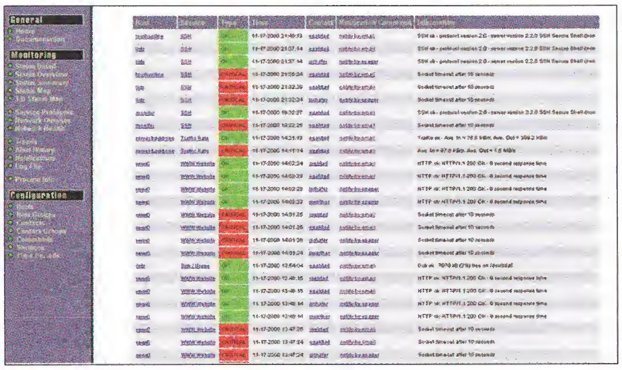
As you can see from the figure, NetSaint provides links within each table to more detailed information. A report for a particular host, for example, might indicate that a Microsoft Windows system was down for over two hours in a five-hour period. A system administrator can provide a comment, such as "swapped out disk drive," to explain any unusual circumstances. The report also displays information about the host's recent history and its current monitoring configuration.

Figure One also shows the NetSaint menu bar in the window's leftmost frame. The items listed under "Monitoring" select various status displays. "Status Detail" displays overall status in tabular form; "Status Overview" shows a breakdown of host and service status by the devices' physical location (the accounting department, for instance) or by how they're used (printers, for instance). In this way, the location of trouble can be determined quickly.

NetSaint's Host Commands menu, shown in *Figure Two* (pg. 24), lets the administrator change numerous aspects of a host's monitoring configuration, including enabling and disabling monitoring and/or alert notifications, adding and modifying scheduled downtime for the host (during which monitoring ceases and alerts are not sent), and forcing all defined checks to be run immediately (rather than waiting for their next scheduled instance).

The second menu item allows you to acknowledge any

FIGURE ONE: The NetSaint Network Monitor



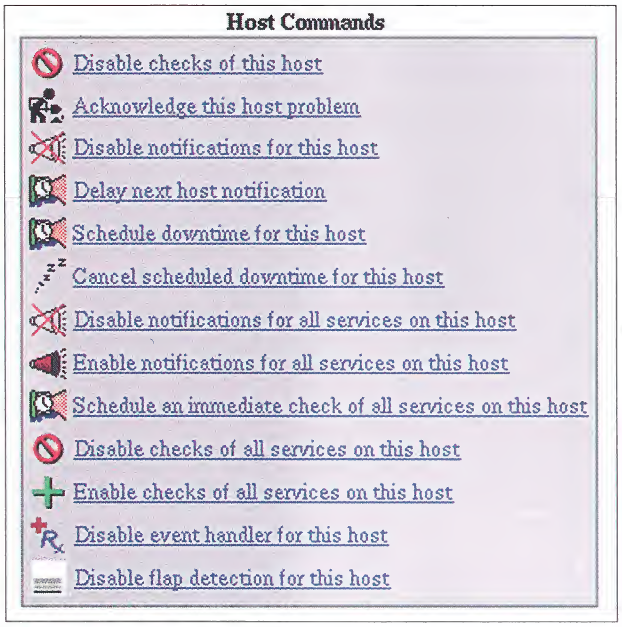

Host Commands
Disable checks of this host
Acknowledge this host problem
Disable notifications for this host
Delay next host notification
Schedule downtime for this host
Cancel scheduled downtime for this host
Disable notifications for all services on this host
Enable notifications for all services on this host
Schedule an immediate check of all services on this host
Disable checks of all services on this host
Enable checks of all services on this host
Disable event handler for this host
Disable flap detection for this host

FIGURE TWO: NetSaint's Host Menu

current problem. Acknowledging a problem simply means "I know about the problem, and it's being handled." NetSaint marks the corresponding event as such, and future alerts are suppressed until the item returns to its normal state. This process also allows you to enter a comment explaining the situation, which is helpful when more than one administrator is examining the monitoring data.

CANONIZE YOUR OWN NETSAINT

Configuring NetSaint can seem daunting at first, but it's actually quite straightforward once you understand all the pieces. It has several configuration files:

- The *netsaint.cfg* file configures the NetSaint daemon: you can specify parameters such as the directory locations for the package's various components, the user and group ids for the *netsaint* daemon, which items to log, log file rotation settings, timeouts and other performance-related settings. The file also contains additional settings for advanced features like event handling and notifications.

- The *commands.cfg* and *hosts.cfg* files define the commands that will test the various hosts and network services being monitored. By convention, *commands.cfg* defines commands and *hosts.cfg* defines hosts and services.

- The *nscgi.cfg* file configures the NetSaint displays, including paths to Web page items and scripts, and per-item icon image and sound selections. It also authorizes access to NetSaint's data and commands.

- The *resource.cfg* file defines macros that may be used within other settings for clarity and security purposes (e.g., to hide passwords from view).

NetSaint has many nice features. First of all, it can save data between runs (this is the default configuration) so you analyze trends over time (see the accompanying article "Identifying and Analyzing Trends," pg. 29). You can also specify whether or not to display the saved status information when the NetSaint page is first opened. The following *netsaint.cfg* entries control the latter feature:

```
retain_state_information=1
retention_update_interval=60
use_retained_program_state=1
```

You can also save the data produced by the status commands for future use outside of NetSaint, using these *netsaint.cfg* entries:

```
process_performance_data=1
service_perfdata_command=commandname
```

The arbitrary *commandname* specified in the second entry is defined in *commands.cfg*. A typical entry might write the status commands' output to an external file:

```
command[commandname]=/bin/echo $OUTPUT$
>>filename
```

The macro `$OUTPUT$` is replaced with the entire output returned by the status commands. Once captured, the data in the external file can be sent to a database or processed in any way that you like. We'll talk more about the syntax of *commands.cfg* shortly.

The NetSaint daemon can also be configured to accept data from outside sources. In this passive mode, the remote device performs a check and writes the result in a predetermined format to a file on the NetSaint host machine. This mode may be enabled by inserting the following line into the *netsaint.cfg* file:

FIGURE THREE: Sample nscgi.cfg Entries

```
use_authentication=1
authorized_for_configuration_information=netsaintadmin,root,chavez
authorized_for_all_services=netsaintadmin,root,chavez,maresca
hostextinfo[bagel]=;redhat.gif;;redhat.gd2;;168,36;,,;
```

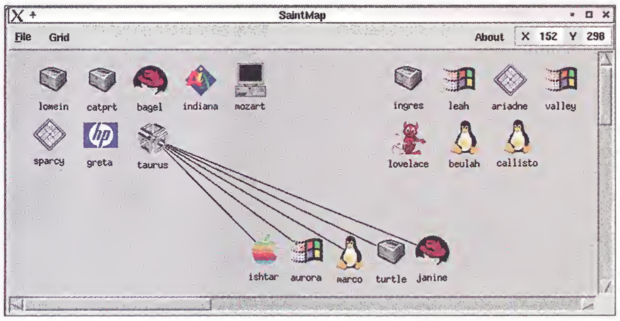


FIGURE FOUR: A NetSaint Status Map

```
check_external_commands=1
```

Next, you might want to take a look at the *nscgi.cfg* file, which defines who can access NetSaint and what they can do. *Figure Three* shows some sample entries from that file. The first entry enables the access control mechanism. The next two entries specify the users who are allowed to view NetSaint's configuration information and the status of the services NetSaint is monitoring. All users must also be authenticated by the Web server using the Apache *htpasswd* mechanism.

The final entry in *Figure Three* specifies extended attributes for the host named `bagel`. The file names in this example specify image files for the host in the status tables (GIF format) and in the status map (GD2 format); the two numeric values specify the device's location within the status map (consult the man page for complete details). NetSaint's status maps provide a quick way of accessing information about individual devices.

A sample status map is displayed in *Figure Four*. The map was created with David Kmoch's *saintmap* utility (http://netsaint.org/download), which provides a convenient way to create status maps. In this case, we've grouped devices by their physical location (although we haven't bothered to label the groups). The lines from `taurus` to each device in the bottom group indicate that `taurus` is the gateway between our NetSaint host and this location. When the map is used by NetSaint, each icon will have a status indication (up or down) added to it, enabling an administrator to get an overall view of things right away, even when the network is very large and complex.

NETSAINT NUTS AND BOLTS

Let's look in more detail at the syntax of two important NetSaint files, *commands.cfg* and *hosts.cfg*. By convention, *commands.cfg* defines monitoring tasks and associated commands, whereas *hosts.cfg* defines hosts and services.

Figure Five shows a sample *commands.cfg* file. It creates two monitoring tasks, one named `do_ping` and another called `check_telnet`. *Figure Six* lists a sample *hosts.cfg* file; it defines two hosts, `ishtar` and `callisto`. Finally, *Figure Seven* shows some sample services.

Note that a service combines a `host` and a `command`. To use NetSaint, you define your hosts, create a pool of monitoring tasks, and then tie tasks to a host. For example, `host[callisto]` defines a device named "callisto"; `command[check_telnet]` defines a command called

FIGURE FIVE: Sample Command Definitions

```
# command entries define a monitoring task and its associated command.
# command entries are also used to define commands used for other purposes like
# sending alerts and event handlers.
command[do_ping]=/bin/ping -c 1 $HOSTADDRESS$
command[check_telnet]=/usr/local/netsaint/libexec/check_tcp -H $HOSTADDRESS$ -p 23
```

FIGURE SIX: Sample Host Definitions

```
# host: define a host/device to be monitored.
# hostgroup: create a list of hosts to be grouped together in displays.
# service: define an item on a host/device to be checked periodically.
# contact: specify a list of recipients for alerts.
# timeperiod: assign a name to a specified time period.
#
#host[label]=description;IP address;parent;check command;;;;;;
host[ishtar]=ishtar (with printer);192.0.76.98;taurus;check-printer-
alive;10;120;24x7;1;1;1;
host[callisto]=callisto;192.0.22.124;;check-host-alive;10;120;24x7;1;1;1;
```

FIGURE SEVEN: Sample Service Definitions

```
#service[host-label]=service-label;; when;;;; notify;;;;;; check-command
service[callisto]=TELNET;0;24x7;4;5;1;admins;960;24x7;0;0;0;;check_telnet
service[callisto]=PROCS;0;24x7;4;5;1;admins;960;24x7;0;0;0;;snmp_nproc!com
mune!250!400
service[ingres]=HPJD;0;24x7;4;5;1;locals;960;24x7;0;0;0;;check_hpjd
```

"check_telnet"; they are tied together in a definition like `service[callisto]=TELNET ... check_telnet` which tests the telnet capabilities of the device named callisto using the command `check_telnet`. Let's look at the contents of each file in more detail.

CREATING COMMANDS

The first `command` entry in *Figure Five* defines a command called `do_ping` that runs the *ping* utility to send a single ICMP Echo packet to an IP address. The IP address is not defined; instead the built-in macro `$HOSTADDRESS$` will be replaced with the IP address of the host that we want to *ping*.

The second entry in *Figure Five* defines a `check_telnet` command, which runs the NetSaint plug-in named `check_tcp`; `check_tcp` attempts to connect to the TCP port specified by `-p` on the indicated host. In this case, it's port 23, the standard Telnet port. Again, this entry uses `$HOSTADDRESS$` instead of a fixed IP address.

In addition to these macros, `command` entries can also use arguments to parameterize commands. Here's an example:

```
command[check_tcp]=
/usr/local/netsaint/libexec/check_tcp -H
$HOSTADDRESS$ -p $ARG1$
```

This entry defines the `check_tcp` command differently than the one shown in *Figure Five*. If this `check_tcp` is used in a `service` entry, NetSaint calls the same plug-in as in the earlier `check_telnet` command, but uses the first argument passed in the `service` entry as the desired port number (set by the `-p` flag). Macros of the form `$ARGn$` allow you to use any number of arguments in a command.

Many NetSaint plug-ins also use the `-w` and `-c` options to define value ranges that should generate warning alerts and critical alerts, respectively. Somewhat counterintuitively, the argument to these options specifies the range which should *not* be considered problematic.

For example, the following entry defines the command `snmp_load5`, setting the warning level to values over 150, and the critical level to over 300:

```
command[snmp_load5]=
/usr/local/netsaint/libexec/check_snmp
-H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.2021.10.1.5.2
-w 0:150 -c :300 -l load5
```

This entry calls the `check_snmp` command for the current host, using the first command argument as the SNMP community name (the `-C` flag), and retrieves the 5-minute load average value (multiplied by 100, as specified by the SNMP label in the -o option), labeling the data as `load5`. The value will trigger a warning alert if it is over 150; `-w 0:150` means that values from 0 to 150 inclusive are not in the warning range. It will also trigger a critical alert if it is over 300, i.e., not in the range 0 (used as the default if there's no value before the colon) to 300 inclusive. If a value would trigger both a warning alert and critical alert, only a critical alert is issued.

DEFINING HOSTS

Figure Six illustrates the definitions for hosts. Let's take these entries apart, field by field. Individual fields are separated by semicolons.

The first field is the most complicated and has the following syntax: `host[`*label*`]=`*description*, where *label* is the label to be used in status displays and *description* is a (possibly longer) phrase describing the device.

The second field holds the device's IP address. This field actually identifies the desired device (the preceding items are just arbitrary labels).

The third field specifies the parent device for the item: a list of one or more labels for intermediate devices located between the current system and this one. For example, to reach `ishtar`, we must go through the device named `taurus`, and so `taurus` is specified as its parent. The parent is optional, and the entry for `callisto` does not use it.

The fourth field specifies the command NetSaint should use to determine whether the host is accessible ("alive") or not. (The two used here, `check-printer-alive` and `check-host-alive` come with the standard NetSaint package.)

The fifth field indicates how many checks must fail before the host is assumed to be down (10 in our examples).

The remaining fields in the example entries are used to customize alert notifications.

Sixth field: the time interval between alerts when a host remains down, in minutes (here, 120 minutes or 2 hours).

Seventh field: the time period during which alerts should be sent. The time period is defined elsewhere in the configuration file as a `timeperiod` entry. This one, named "24x7," is pre-defined to mean "all the time." It's a convenient choice when you are getting started using NetSaint.

FIGURE EIGHT: Sample Event Handler

```
#event handler for disk full failures
command[clean]=/usr/local/netsaint/local/clean $STATETYPE$
service[beulah]=DISK;0;24x7;4;5;1;locals;960;24x7;0;0;0;clean;check_disk!/!15!5
```

Eighth field: a flag (0 means no; 1 means yes) whether to notify when the host recovers after being down.

Ninth field: flag indicating whether to notify when the host goes down.

Tenth field: flag indicating whether to notify when the host is unreachable due to a failure of an intermediate device.

All the flags are set to yes in our examples.

MONITORING SERVICES

Now that we have defined entries for both commands and hosts, we are ready to define specific items that NetSaint should monitor. These items are known as "services" and are created using the `service` entry. Examples are shown in *Figure Seven*.

The most important fields in these entries are the first, third, seventh, and final ones, which hold the following settings:

- The service definition (field 1), using the syntax `service`[*host-label*]=*service-label*. For example, the first entry in *Figure Seven* defines a service named `TELNET` for the host entry labeled `callisto`.

- The name of the time period during which this check should be performed (field 3), again defined in a `timeperiod` entry.

- The contact name (field 7): this item holds the name of a contact entry defined elsewhere in the file. The `contact` entry type is used to specify lists of users to be contacted when alerts are generated.

- The command to run to perform the check (final field), defined via a `command` entry elsewhere in the configuration file. Arguments to the command are specified as exclamation-mark-separated subfields with the command (as in the `PROCS` example service). These will be passed through to the `$ARG`*n*`$` macros in the command definitions.

The other fields hold the volatility flag (field 2), the maximum number of checks before a service is considered down (4), the number of minutes between normal checks (5) and failure rechecks (6), the number of minutes between failure alerts while the service remains down (8), the time period during which to send alerts (9), three alert flags (10-12) corresponding to service recovery, and whether or not to send critical alerts and warning alerts, respectively. The next-to-last field (13) holds the command name for the event handler for this service (see below); no event handler is specified in these cases. The default values, used in the examples, are good starting points, and these fields are further defined in

NETWORK MONITORING SOLUTIONS

There's no shortage of packages that provide more complex monitoring and event-handling capabilities. While these packages can be very powerful tools for information gathering, their installation and configuration complexity typically grows as the number of features does.

OPEN SOURCE SOFTWARE

There are many Open Source monitoring programs and projects, including OpenNMS (http://www.opennms.com), Sean MacGuire's Big Brother (http://www.bb4.com), and Thomas Aeby's Big Sister (http://bigsister.graeff.com). A more basic Open Source program, the Angel Network Monitor by Marco Paganini (http://www.paganini.net/angel), is also available.

COMMERCIAL SOFTWARE

If you can afford it, there's nothing like a full-scale commercial network monitoring package for enterprise-wide monitoring. The amount of time to set up and learn to use one tends to be less than that of an Open Source solution. These days, all major commercial products can monitor Linux systems, and some even allow a Linux system to be the master monitoring station.

You should expect most or all of the following features from a commercialnetwork monitoring product:

- Excellent performance, even for large networks. One disadvantage of the Open Source packages is that they generally rely on scripts for data collection (at least in part), an approach that tends to not scale well beyond a certain point (reached by a medium-sized network). Commercial programs can use the performance advantages of compiled code (as opposed to interpreted scripts).

- Device autodetection: commercial packages will search for and detect new devices that appear on the network and automatically add them to the set being monitored.

- Support for heterogeneous networks, including mixed UNIX and Windows environments. Some of the Open Source packages also support such environments.

- Multiple server-based monitoring, designed to distribute the overhead load somewhat and also prevent a single point of failure.

- More sophisticated authentication techniques and data protection.

- Built-in report generation and graphing facilities.

The best known of the commercial products are OpenView from Hewlett-Packard, Patrol from BMC, Tivoli from IBM, and Unicenter from Computer Associates; Unicenter won the "Best in Show" award at the January 2002 LinuxWorld Expo.

the NetSaint documentation.

NetSaint displays can summarize status information for a group of devices. You specify this by defining a `hostgroup`. For example, the following configuration file entry defines the `Printers` group:

```
hostgroup[Printers]=
 Printers;locals;
 ingres,lomein,turtle,catprt
```

The syntax is simple: `hostgroup`[*group-label*]=*description*; *contact-group*;*list-of-host-labels*. Keep in mind that the host labels refer to the names of host definitions within the NetSaint configuration file and not necessarily to literal hostnames. The members of the specified contact group will be notified whenever there is a problem with any device in the list.

CREATING EVENT HANDLERS AND ALERTS

In addition to sending alert messages, NetSaint also provides support for event handlers: commands to be performed when a service check fails. In this way, you can have NetSaint begin to deal with a problem before you even know about it. *Figure Eight* shows the entries which correspond to a simple event handler.

First, we define a command named `clean` which specifies a script to run. Its sole argument is the value of the `$STATE-TYPE$` NetSaint macro, which is set to `"HARD"` for critical failures and `"SOFT"` for warnings. The `clean` command is then specified as the event handler for the `DISK` service on `beulah`. (In this example, the `/usr/local/netsaint/local/clean` script is one we custom-wrote to use the *find* command to delete junk files within the root filesystem and use its argument's value to decide how aggressive to be.)

The three arguments to `check_disk` are the file system to check (`/`), the amount of free space there must be in order to not trigger a warning (`15`, so a warning is sent if the file system is more than 85% full), and the amount to not trigger a critical alert (`5`, meaning a crtitical alert is sent if the file system is more than 95% full).

PLAN AHEAD

Network monitoring software can be a powerful tool for keeping track of system status, both in real-time and over the long haul. However, don't underestimate the time it will take to implement a monitoring strategy in the real world. Careful planning can minimize the amount of time that it will require, but it's always a big job. You need to consider not only the installation and configuration issues but also the performance impact on your network and the security ramifications of the daemons and protocols you're enabling. While this can be a daunting task that cannot be rushed, in the end it's worth the effort.

Æleen Frisch is the author of Essential System Administration and writes the "Guru Guidance" column for Linux Magazine. She can be reached at aefrisch@lorentzian.com.

RESOURCES

NetSaint
http://netsaint.org/

SaintMap
http://netsaint.org/download/

OpenNMS
http://www.opennms.com/

Big Brother (free for non-commercial uses)
http://www.bb4.com/

Big Sister
http://bigsister.graeff.com/

Angel Network Monitor
http://www.paganini.net/angel/

Multi-Router Traffic Grapher (MRTG)
http://www.mrtg.org/

RRdtool
http://rrdtool.eu.org

Cricket
http://cricket.sourceforge.net/

James Moore's Cricket Add-ons
http://www.certaintysolutions.com/tech-advice/cricket-contrib/

RRGrapher
http://net.doit.wisc.edu/~plonka/RRGrapher/

OpenView
http://www.openview.hp.com/

Patrol
http://www.bmc.com/patrol/

Tivoli
http://www.tivoli.com/

Unicenter
http://www.cai.com/unicenter/

Additional information about programs mentioned in this article can be found online at http://www.linux-mag.org/downloads/2002-05/netmonitor.tgz.

Identifying and Analyzing **Networking Trends**

By Æleen Frisch

NetSaint is very good about providing up-to-the-minute status information, but there are also times when it's helpful to compare current data with past values. This is essential for performance tuning and capacity planning.

One of the best-known packages of this type is the Multi-Router Traffic Grapher (MRTG, http://www.mrtg.org), written by Tobi Oetiker and Dave Rand. It collects data and automatically produces graphs of these values over various time periods. As its name suggests, it was designed to track the performance of the routers in a network, but can be used for a wide variety of data — in fact, for any value that can be tracked over time.

When a new data point is added, the oldest is deleted from storage, and the result is referred to as "round-robin data" (RRD). Summary values are also stored over various time periods. This strategy results in small, fixed-size databases that nevertheless offer a wealth of important information.

More recently, MRTG has been supplanted by Oetiker's newer package, RRDtool (http://rrdtool.eu.org), which has much more powerful and configurable graphing facilities. It does, however, require a separate data collection script or package, such as the popular Cricket (http://cricket.sourceforge.net).

Figure A shows a command to create a database named `ping.rrd` consisting of two fields, `trip` and `lost`, which are defined by the two data set (`DS`) lines. These fields hold the round-trip travel time and the percentage of lost packets resulting from running the `ping` command. Both fields are of type `GAUGE`, meaning that the values should be interpreted as separate values. Alternatives to the `GAUGE` type are referred to as counters, and their values are interpreted as changes with respect to the preceding value; they include `COUNTER` for monotonically increasing data and `DERIVE` for data which can vary up or down.

The fourth field in each `DS` line is the time period between data samples, in seconds (here 10 minutes), and the final two fields hold the valid range of the data. A setting of `U` stands for "unknown", and two of them have the effect of allowing the data itself to define the valid range (i.e., accept any value).

The remaining lines of the command, labeled `RRA`, create round-robin archive data within the database. The second field indicates the kind of aggregate value to compute: here, we compute averages and maximums. The remaining fields specify the maximum percentage of the required sampled data that can be missing (50%), the number of raw values to combine and the number of data points of this type to store.

Those final two fields can be confusing at first. For example, values of 6 and 700 mean that the average (or other function) of six raw values will be computed, and the most recent 700

FIGURE A: Creating a Database with RRDtool

```
rrdtool create ping.rrd \
 —step 300 \
 DS:trip:GAUGE:600:U:U \
 DS:lost:GAUGE:600:U:U \
 RRA:AVERAGE:0.5:1:600 \
 RRA:AVERAGE:0.5:6:700 \
 RRA:AVERAGE:0.5:24:775 \
 RRA:AVERAGE:0.5:288:797 \
 RRA:MAX:0.5:1:600 \
 RRA:MAX:0.5:6:700 \
 RRA:MAX:0.5:24:775 \
 RRA:MAX:0.5:288:797
```

FIGURE B: Adding Data Using RRDtool

```
#!/bin/csh

ping -w 30 -c 10 $1 > /tmp/ping_$1
set trip=`tail -1 /tmp/ping_$1 | awk -F= '{print $2}' | awk -F/ '{print $2}'`
set lost=`grep transmitted /tmp/ping_$1 | awk -F, '{print $3}' | awk -F% '{print $1}'`
rm -f /tmp/ping_$1
rrdtool update ping.rrd "N:"$trip":"$lost
```

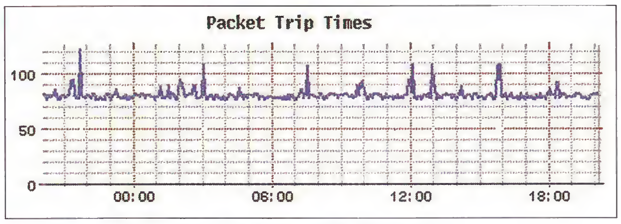
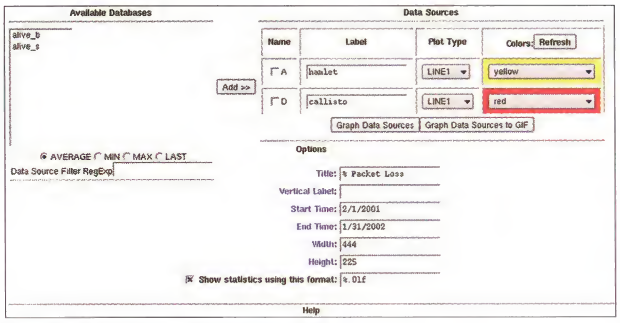


FIGURE C: Creating a Chart With RRDtool



alive_b
alive_s

[✓ AVERAGE] MIN MAX LAST

Data Source Filter RegExp:

Name	Label	Plot Type	Colors
A	hamlet	LINE1	yellow
D	callisto	LINE1	red

Title: % Packet Loss
Vertical Label:
Start Time: 2/1/2001
End Time: 1/31/2002
Width: 444
Height: 225
[✓] Show statistics using this format: %.01f

FIGURE D: The RRGrapher Utility

averages will be saved. The default time period between data points is 300 seconds (this can be changed by using the `—step` option). Thus, the second `RRA` line will calculate a 30-minute average value (6 * 5 minutes), and we will have them going back for 350 hours (700 * 6 * 5 minutes).

Thus, in our example database, we are creating five-minute averages and maximums, 30-minute averages and maximums, 2-hour averages and maximums (24 * 5 minutes) and daily averages and maximums (288 * 5 minutes = 1 day). We will have five-minute data going back for 50 hours (600 * 5 minutes), 30-minute data for 350 hours (700 * .5 hours), 2-hour data for 1,550 hours (775 * 2 hours), and daily data for over 2 years (797 days).

There are many ways to add data to an RRDtool database. *Figure B* (pg. 29) shows a script illustrating one of the simplest: using the `update` argument.

After the `ping` command generates the data, the next two lines take apart the output and store the results in two variables. The command `rrdtool update` then adds it into the database. The final argument to the `rrdtool` command is a colon-separated list of data values, beginning with the time to be associated with the data (`N` means "now"), followed by the value for each defined data field, in order.

In this case, we used normal UNIX commands to obtain the data we needed, but we could also have used SNMP as the source. For more on SNMP, the Simple Network Management Protocol, see "Network Device Interrogation" in the November 2001 issue (available online at http://www.linux-mag.com/2001-11/snmp_01.html) and in this month's Guru Guidance column (available online at http://www.linux-mag.com/2002-05/guru_01.html).

Once we've accumulated data for a while, we can begin to create graphs. We'll again use RRDtool. For example, the following command creates a simple graph of the data:

```
rrdtool graph ping.gif \
 —title "Packet Trip Times" \
 DEF:time=ping.rrd:trip:AVERAGE \
 LINE2:time\#0000FF
```

This command defines a graph of a single value, specified via the `DEF` (definition) line. The graphed variable is named `time` and it comes from the stored averages of the `trip` field in the `ping.rrd` database (raw values cannot be graphed). The `LINE2` command is what actually graphs the values. `LINE2` refers to a fever chart of the defined variable (`time`), displayed in the color corresponding to the RGB value `#0000FF` (blue). The result is seen in *Figure* C. The backslash before the number sign is required to protect it from the shell, but is not part of the command syntax. The resulting graph is placed in the file `ping.gif`.

Creating graphs like these can be tedious. Fortunately, there is RRGrapher (http://net.doit.wisc.edu/~plonka/RRGrapher/), a utility that automates the process. This CGI script, written by Dave Plonka, is illustrated in *Figure D*.

You can use this tool to create graphs which draw data from multiple RRD databases. In this figure, we're plotting values from two databases over a specified time period. The latter is one of RRGrapher's most convenient features, since RRDtool requires times to be expressed in standard UNIX format (seconds since January 1, 1970) but you can enter them here in a readable format.

As mentioned earlier, you'll want a front-end package, such as Cricket, to automate the process of gathering data for

FIGURE E: The cricket-config/hosts/callisto/Targets config file

```
Target —default-
   server               =       callisto
   snmp-community       =       somesecret

# Specify data source groups to collect
target ucd_sys
   target-type          =       ucd_System
   short-desc           =       "CPU, Memory, and Load"

target boot
   target-type          =       ucd_Storage
   inst                 =       1
   short-desc           =       "Bytes used on /boot"
   max-size             =       19487
   storage              =       boot
```

RRDtool. Cricket is written in Perl, and requires a very large number of modules to function (plan on several visits to CPAN), so installing it may take a bit of time. Once it's up and running, its most important components are:

- The *cricket-config* directory tree, which contains specifications for each device to be monitored.
- The *collector* script, which runs periodically from *cron* (usually, every 5 minutes).
- The *grapher.cgi* script, which is used to display Cricket graphs within a Web browser.

The *cricket-config* directory tree contains the configuration files that tell the *collector* script what data to get from which devices. It holds a hierarchical set of configuration files. Default values set at each level continue to apply to lower levels unless they are explicitly overridden. Once the initial setup is completed, adding additional devices is simple.

The first-level subdirectories within this tree refer to broad classes of devices: routers, switches, and so on. We'll see an example of the device class `hosts`, provided as part of the Unix Host Resources package written as a Cricket add-on by James Moorei for devices that support SNMP (http://www.certaintysolutions.com/tech-advice/cricket-contrib/).

The file *cricket-config/hosts/Defaults* supplies default values for entries within the *hosts* subtree. The *Defaults* file, for instance, defines a data set called `ucd_System` that includes CPU and memory usage, as well as load averages over 1, 5, and 15 minutes. The `ucd_Storage` data set includes information on disk usage.

Every host to be monitored has a subdirectory under *hosts* that contains a file named *Targets*, which contains the information necessary to build a round-robin data set for that particular host. *Figure E* shows some excerpts from the file for the host `callisto`.

This file instructs Cricket to collect values for all of the items defined in the `ucd_System` and `ucd_Storage` groups. Each target will appear as an option within the Cricket Web interface for this host.

Graphs of the Cricket data can be very helpful in determining what the normal range of behavior is for various devices. When you understand the normal status and variation, you are in a much better position to recognize and understand the significance of anomalies that turn up.

Æleen Frisch is the author of Essential System Administration and writes the "Guru Guidance" column for Linux Magazine. She can be reached at aefrisch@lorentzian.com.

ACCELER8 YOUR SYSTEM

NEW! WITH KDE 3.0!

8 REASONS FOR USING SUSE LINUX:

8.0 KDE 3.0 - the easiest desktop interface
8.0 Extensive and automatic hardware detection
8.0 Multimedia applications - digital sound and video (ALSA 0.9)
8.0 StarOffice - the MS-compatible office system
8.0 Personal firewall security for your system
8.0 YaST2 - simple system and network configuration
8.0 SuSE YOU - instant online update facility
8.0 Stability, Reliability - Open Standards!

Please bill the following address:

Company ____
Name ____
Street ____
Postal-Code ____ Zip-Code ____
E-mail ____ Phone ____

☐ Personal 8.0 $39.95
☐ Professional 8.0 $79.95

Order online today! http://shop.suse.com/lm

info@suse.com (888) 875-4689 (510) 628-3381

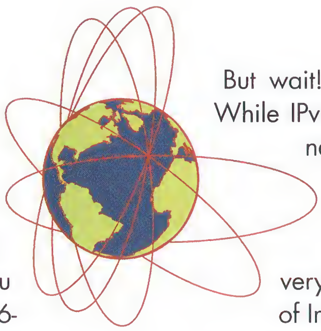
SuSE Inc. — The Linux Experts
318 Harrison Street, Suite 301
Oakland · CA 94607

SuSE

The Future of INTERNET ADDRESSES

IPv6 Aims to Save the World from a Looming IP Address Shortage

by Peter Bieringer



So you've heard about Internet Protocol Version 6 (IPv6), the latest and greatest Internet Protocol address strategy. You've heard that it will solve the looming shortage of IP addresses, that it changes the format of IP addresses, and even that some sites are using IPv6 now. You might be wondering if your Linux system is IPv6-ready, whether you can use IPv6 now, and whether you can set up your Apache Web server to use it. For some versions of Linux the answers to these pressing questions are yes, yes, and yes.

But wait! Don't run off and log on as root yet! While IPv6 is largely defined and some Linux kernels support it, a variety of issues remain unresolved and some features are not yet implemented. IPv6 is not ready for prime-time yet. Having said that, it's still very worthwhile to look at the next generation of Internet addressing.

Let's take a closer look at how IPv6 differs from its predecessor, IPv4, walk through the process of enabling IPv6 on your Linux system, and configure an Apache Web server to recognize the new IP address formats.

FROM IPv4 TO IPv6

An Internet Protocol (IP) address uniquely identifies a single device connected to the Internet. In Internet Protocol Version 4 (IPv4) an IP address is 32 bits long and is represented visually by separating the address into four octets — four groups of eight bits represented in decimal notation — separated by dots. Familiar IP addresses, such as `127.0.0.1`, are based on the IPv4 specification.

Since the late 1970s, when IPv4 was designed, the number of computers connected to the Internet increased from 213 (in August 1981) to 125 million (in June 2001).

IPv6 (sometimes called "IP next generation" or IPng) has been in development since 1990 and offers several substantial improvements over IPv4. Briefly, it supports roaming between different networks (necessary for mobile devices), authentication, and encryption and compression of IP traffic. It allows devices to calculate an IP address automatically (yes, you'll never have to manually assign IP addresses again).

Of course, IPv6 also provides a much larger set of IP addresses. In fact, when adopted, IPv6 will allow *every* connected device to have a unique and directly addressable IP address. Network Address Translation will be gone for good.

To provide that many unique addresses, IPv6 increases IP address size to 128 bits from 32 bits. A 128-bit IPv6 address looks like this:

```
3ffe:ffff:0100:0000:0000:0000:0000:0001
```

IPv6 switches from decimal to hexadecimal notation, uses groups of 16 bits instead of 8 bits, and uses a colon rather than a dot as a separator.

The example address shown above contains a lot of zeroes. You can drop leading zeroes in any block, so the address above simplifies to:

```
3ffe:ffff:100:0:0:0:0:1
```

To further simplify things, one (and only one) sequence of blocks containing only zeroes can be dropped. Now our address looks like:

```
3ffe:ffff:100::1
```

The special address `any`, which matches any legal address and is represented as `0.0.0.0` in IPv4, is the address `0000:0000:0000:0000:0000:0000:0000:0000` in IPv6, but is often compressed to just "`::`". The IPv6 `localhost` address is `0000:0000:0000:0000:0000:0000:0000:0001`, and it compresses to "`::1`".

NETMASKS AND ROUTING

The Internet is a network of sub-networks (or *subnets*) where each subnet can contain (smaller) subnets and every device is connected to at least one subnet. IP addresses reflect this structure. One portion of the IP address, the netbits, identifies the sub-network a device is on. The other portion, the hostbits, identifies a specific device connected to that subnet. (Netbits are the most significant, or leftmost, bits of an IP address. Hostbits are the least significant, or rightmost, bits.) In IPv4 the netmask describes how many bits are in the netbits and hostbits portions of an address. IP addresses, the netmask, and the default router on each subnet work together to correctly process, or route, messages from one machine to another.

A typical IPv4 netmask might be `255.255.255.0` (24 netbits to identify the subnet and 8 hostbits to identify the machine). While effective, the IPv4 netmask notation is unwieldy. For IPv6, the protocol designers chose the Classless Inter-Domain Routing (CIDR) schema to specify how many bits in the Internet Protocol address to use for routing. A netmask of `255.255.255.0` becomes `/24` under CIDR, while `255.0.0.0` becomes `/8`. The IPv4 notation for network `10.0.0.0` with a netmask of `255.0.0.0` translates to `10.0.0.0/8`. (CIDR is also called the "slash" notation.)

In IPv6 the number after the slash is called the "prefix length." An example looks like this:

```
3ffe:ffff:100::/64
```

The netbits or prefix is the most significant 64 bits. The remaining part of the address (64 bits, in this case) is the suffix. In IPv4 terms, the prefix represents the network part, while the suffix represents the host part. The general numbering scheme for IPv6 is described in the IPv6 HOWTO (http://linuxdoc.org/HOWTO/Linux+IPv6-HOWTO/).

IPv6 IN LINUX

IPv6 was first implemented in the Linux kernel in version 2.1.8 (November 1996). Development today is mostly done by the USAGI project in Japan (http://www.linux-ipv6.org/). USAGI started in October 2000 with the goal of modifying a vanilla kernel to include more IPv6-related features. Because IPv6 is still in flux, most of the work of the USAGI team is currently available only as patches for the 2.2 or 2.4 series kernels. However, some core features that are well-defined and stable are widely available.

It's easy to see if your Linux system is able to use IPv6 networking. If you're running a 2.2 series kernel, installed by older distributions, it's not likely that the kernel is IPv6-enabled. It's possible to recompile your kernel (using the USAGI patches), but that is beyond the scope of this article. You can find pointers in the HOWTO.

FIGURE ONE: Output From *ip* Command

```
# /sbin/ip -6 addr show eth0

3: eth0:  <BROADCAST,NOTRAILERS,UP> mtu 1500 qdisc pfifo_fast qlen 100
           inet6 fe80::212:34ff:fe12:3456/10 scope link
```

If you're using a series 2.4 kernel, IPv6 is probably already included or is available as a module. To see if your current kernel supports IPv6, simply issue the command

```
# ls /proc/net/if_inet6
```

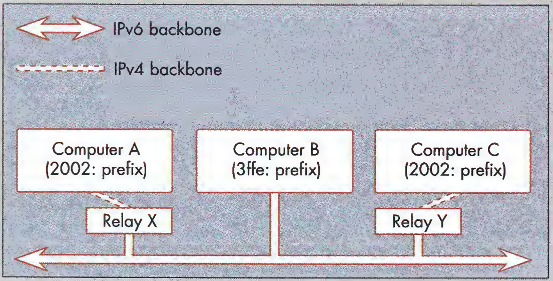
If there's no error, you're set. If you don't find this file on your system, then IPv6 support is missing. You can load the IPv6 module by running the command

```
# modprobe ipv6
```

as root. If no error is reported, the module is loaded. (You can

FIGURE TWO: Connect to the IPv6 Backbone

Computers A and C use 6to4 tunnels to connect to the IPv6 backbone; their addresses begin with 2002: Computer B is connected directly to the IPv6 backbone; its address begins with 3ffe:, which indicates a 6bone testbed address.


IPv6 backbone
IPv4 backbone
Computer A (2002: prefix)
Computer B (3ffe: prefix)
Computer C (2002: prefix)
Relay X
Relay Y

check by running `lsmod` as root.) At this point `ls /proc/net/if_inet6` should work, too.

In theory, all network devices should be able to support IPv6; however, some are not able to transport native IPv6 packets because their transport protocol is missing a type field. Devices that cannot support IPv6 include Serial Line IP (SLIP) devices (`sl*`), Parallel Line IP (PLIP) devices (`plip*`), and some ISDN devices (`isdn*` using "rawip" encapsulation and all `ippp*` devices).

TIME TO FLEX YOUR MUSCLES

To use the newest network features, you need the *ip* utility; *ip* replaces the *ifconfig* and *route* utilities used for IPv4. *Figure One* (pg. 33) shows typical output from the `ip` command using the `-6` option for IPv6.

The output of the *ip* command shows that your device is ready. The IPv6 address was derived from your hardware MAC address and has an address scope of `link` (also known as link-local). Packets from a link-local addresses cannot be sent past a router, and generally cannot be used for HTTP, telnet, and the like. Instead, link-local addresses are used for several auto-configuration features built into IPv6, such as:

- Identifying what device has a particular address on the local link and finding its hardware MAC address (this feature replaces *arp* from IPv4).

- Recognizing whether there is a router on the link.

VENTURING FORTH

Now that your device is working, how do you find an Internet Service Provider that handles native IPv6? You probably can't — they're still pretty rare. Fortunately, the worldwide IPv6 backbone was created with a special tunneling mechanism in place, and its testbed, the 6bone (http://www.6bone.net/), is available for you to use.

6to4 tunneling is the quickest way to connect to the worldwide IPv6 network and is also usable with ISPs that assign IPv4 addresses dynamically through PPP or DHCP. An alternative method, static tunneling, is beyond the scope of this article, but further information is available in the HOWTO.

Figure Two (pg. 33) shows how 6to4 tunneling works. Your computer takes outgoing IPv6 packets and encapsulates them as the payload of an IPv4 packet that is sent to the special address 192.88.99.1. Because of the way the IPv4 backbone is configured, packets sent to this address are delivered to the nearest 6to4 relay. These relays have been established to decapsulate 6to4 packets and route them on the IPv6 backbone, as well as

FIGURE THREE: An IPv6 Routing Table

```
# /sbin/ip -6 route list table all
local ::1 via :: dev lo  metric 0  mtu 3924 rtt 375ms
local ::209.81.9.15 via :: dev lo  metric 0  mtu 3924 rtt 375ms
::/96 via :: dev tun6to4  metric 256  mtu 1480 rtt 375ms
local 2002:d151:90f::1 via :: dev lo  metric 0  mtu 3924 rtt 375ms
2002::/16 via ::192.88.99.1 dev tun6to4  metric 1  mtu 1480 rtt 375ms
2002::/16 dev tun6to4  proto kernel  metric 256  mtu 1480 rtt 375ms
2000::/3 via ::192.88.99.1 dev tun6to4  metric 1  mtu 1480 rtt 375ms
local fe80::2e0:98ff:fe07:c6d5 via :: dev lo  metric 0  mtu 3924 rtt 375ms
local fe80::2e0:98ff:fe08:d28 via :: dev lo  metric 0  mtu 3924 rtt 375ms
fe80::/10 dev eth0  proto kernel  metric 256  mtu 1500 rtt 375ms
fe80::/10 dev tun6to4  proto kernel  metric 256  mtu 1480 rtt 375ms
ff00::/8 dev eth0  proto kernel  metric 256  mtu 1500 rtt 375ms
ff00::/8 dev tun6to4  proto kernel  metric 256  mtu 1480 rtt 375ms
default dev eth0  proto kernel  metric 256  mtu 1500 rtt 375ms
```

FIGURE FOUR: Ping6ing the 6bone

```
# ping6 -c 1 www.6bone.net
PING www.6bone.net(www.6bone.net) 56 data bytes
64 bytes from www.6bone.net: icmp_seq=0 hops=60 time=265.128 msec
—- www.6bone.net ping statistics
—- 1 packets transmitted, 1 packets received, 0% packet loss

# traceroute6 www.6bone.net
round-trip min/avg/max/mdev = 265.128/265.128/265.128/0.000 ms
traceroute to 6bone.net (3ffe:b00:c18:1::10) from 2002:d151:90f::1,
      30 hops max, 16 byte packets
 1  swi6T1-T8.ipv6.switch.ch (3ffe:2000:0:40e::22f)
      252.852 ms  323.346 ms *
 2  2002:823b:1ffd:3:200:cff:fe3e:a419 (2002:823b:1ffd:3:200:cff:fe3e:a419)
      273.631 ms *  271.985 ms
 3  3ffe:200:1:40::2 (3ffe:200:1:40::2)  255.163 ms *  254.314 ms
 4  rap.ipv6.viagenie.qc.ca (3ffe:b00:c18:1:290:27ff:fe17:fc0f)
      461.495 ms  463.318 ms  464.784 ms
 5  www.6bone.net (3ffe:b00:c18:1::10)  471.958 ms  463.246 ms
      463.704 ms

# tracepath6 www.6bone.net
1?: [LOCALHOST]                         pmtu 1480
 1:  swi6T1-T8.ipv6.switch.ch                   asymm  3 224.995ms
 2:  2002:823b:1ffd:3:200:cff:fe3e:a419        asymm  1 370.849ms
 3:  3ffe:200:1:40::2                           asymm  4 395.257ms
 4:  rap.ipv6.viagenie.qc.ca                    asymm  3 617.513ms
 5:  www.6bone.net asymm  4 566.452ms reached
     Resume: pmtu 1480 hops 5 back 4
```

handling return trips.

To accomplish this you'll need a unique address on the IPv6 network. (You already have a link-local address, but you'll need a global one, too.)

First you need the network prefix, which for 6to4 tunnels is `2002::/16`.

Next comes the host suffix, which has the remaining 112 bits, or seven blocks of four hexadecimal digits. Within the suffix, the first two blocks are your unique global IPv4 address, represented in hexadecimal.

The following block is for your use to designate as a subnet; it is zero by default. So if your IPv4 address is 209.81.9.15, your IPv6 network address will begin with `2002:d141:0904:0:`. Finally, the last four blocks are available for you to assign within your network (everything on your side of the 6to4 tunnel). Local 6to4 gateways (your computer, in thie case) usually get the manual suffix `::1`.

This results in a fully specified IPv6 address of `2002:d151:90f::1`. Using this global unique IPv6 address (no other host has your IPv4 address, from which it was generated), you can connect to the IPv6 network. Three types of 6to4 connections are possible:

- From 6to4 to non-6to4 (from your computer): The packets go from your computer via IPv4 to the nearest 6to4 relay, which decapsulates the packet and sends it along the IPv6 backbone toward the destination.
- From non-6to4 to 6to4 (to your computer): The IPv6 backbone routes such packets automatically to one of the 6to4 relays, although it might not be the same one you used upstream. This relay encapsulates the packet as the payload of an IPv4 packet with your IPv4 address, which, as you'll recall, is contained in the IPv6 address, and sends it out over the IPv4 backbone toward you.
- Between 6to4 IPv6 addresses: The packets go from your computer via the IPv4 backbone to the nearest 6to4 relay, which decapsulates them and sends them along the IPv6 backbone toward a 6to4 relay close to the destination. That relay then re-encapsulates the packet and sends it out on the IPv4 backbone toward the final destination.

To begin your connection, first create a dedicated tunnel device of type `sit`, which stands for Simple Internet Transition, a set of protocol mechanisms and operational guidelines designed to ease the transition of the Internet to IPv6 with as little disruption as possible. By design, a `sit` interface will encapsulate upstream (outgoing) IPv6 packets and decapsulate downstream (incoming) ones.

In this example you'll call your `sit` device `tun6to4`. The command is:

FIGURE FIVE: Using *tcpdump* with IPv6

The following command-line options to *tcpdump* are very useful to capture more information about a packet:

`-s 512`	capture 512 instead of the default 68 bytes of a packet
`-vv`	really verbose output
`-n`	don't resolve addresses to names (useful if reverse DNS isn't working properly)

Since *tcpdump* can produce a lot of spurious output, you can place filter expressions on the command line to reduce the noise. The most common ones are:

`proto ipv6`	displays only tunneled IPv6-in-IPv4 traffic (as in the examples in this article)
`ip6`	displays all native IPv6 traffic including ICMPv6
`icmp6`	displays only native ICMPv6 traffic
`not port ssh`	used when running tcpdump in a remote Secure Shell (*ssh*) session

FIGURE SIX: Configuring Apache Virtual Hosts

The first type of virtual host listens on an IPv6 address only:

```
Listen [2002:d151:90f::5]:80
<VirtualHost [2002:d151:90f::5]:80>
ServerName ipv6only.yourdomain.com
  # ... more config lines
</VirtualHost>
```

The second type listens on both IPv4 and IPv6:

```
Listen [2002:d151:90f::2]:80
Listen 1.2.3.4:80
<VirtualHost [2002:d151:90f::2]:80 1.2.3.4:80>
ServerName www4-6.yourdomain.com
  # ... more config lines
</VirtualHost>
```

FIGURE SEVEN: Testing the Apache Server

```
# netstat -lnptu |grep "httpd2\W*$"

tcp 0 0 1.2.3.4:80    0.0.0.0:*   LISTEN  12345/httpd2
tcp 0 0 2002:d151:90f::5:80 :::* LISTEN  12345/httpd2
tcp 0 0 2002:d151:90f::2:80 :::* LISTEN  12345/httpd2
```

```
# /sbin/ip tunnel add tun6to4 mode \
 sit ttl 64 remote any local 209.81.9.15
```

(Note that the remote address is any and your local address is your IPv4 address. Also note that the ip command does not use the -6 option.)

Now bring up the tun6to4 tunnel device by running:

```
# /sbin/ip link set dev tun6to4 up
```

Next, configure your local IPv6 address. We already determined the IPv6 address of our example node as 2002:d151:90f::1. Recall that we also need to specify /16 as the prefix length because we're assigning a 6to4 address. The command is:

```
# /sbin/ip -6 addr add \
 2002:d151:90f::1/16 dev tun6to4
```

You'll want to add the default route for all global IPv6 addresses. Currently they all begin with either *2xxx:* or *3xxx:*. The entire group can be described with 2000::/3, which means the first three bits of the address must be 001 binary and the rest can have any value. We want to route these packets into the tunnel through the IPv4 address 192.88.99.1 of the 6to4 relays. The ip -6 command accepts IPv4 addresses through a compatibility notation, and the full command is:

```
# /sbin/ip -6 route add 2000::/3 via \
 ::192.88.99.1 dev tun6to4 metric 1
```

If you're running the 2.2 kernel, you also need to define how to handle the 2002: prefix. (The 2.4 kernel handles this automatically, so the following command is not needed in 2.4.) If you have a 2.2 kernel, run the command:

```
# /sbin/ip -6 route add 2002::/16 via \
 ::192.88.99.1 dev tun6to4 metric 1
```

Let's take a look at the state of the tun6to4 device by running the following command:

```
# /sbin/ip -6 addr show tun6to4
4: tun6to4@NONE: <NOARP,UP> mtu 1480 qdisc
                 noqueue
   inet6 ::209.81.9.15/128 scope global
   inet6 2002:d151:90f::1/16 scope global
```

You can also check the configuration by examining your routing table with the command ip -6 route list table all, as shown in *Figure Three* (pg. 34). (You might notice tun6to4 is not associated with eth0. That's okay, because the packets are going out on the eth0 interface through IPv4, and eth0 was already configured at boot time through the standard IPv4 tools.)

OTHER IPv6 FEATURES

FIREWALLING

The Linux 2.4 kernel series supports IPv6 firewalling by default (it's maintained by the Netfilter project, http://www.netfilter.org/). The implementation is a little bit behind the IPv4 version, but is already able to protect your host or LAN against unwanted IPv6 traffic. Consider firewalling as soon as possible because unlike on IPv4 networks where internal hosts can be protected automatically by using private IPv4 addresses, IPv6 offers no such protection and a bad guy can connect to your internal IPv6-enabled LAN. Sure, it's not easy to scan the address space (with its maximum 2^{64} addresses per link) in a short time, but commonly used suffixes like ::1 are simple to test. Security is only as strong as its weakest point.

Also, don't just rely on an IPv4 firewall! That's because 6to4 tunneling basically pierces an IPv4 firewall since it cannot decapsulate the packets to see if something is amiss. Check your system by running

```
# netstat -A inet6 -nlptu
```

after an IPv6-enabled restart to see which services are already using IPv6 addresses and are candidates for protection.

As of this writing, IPv6 stateless packet filtering is working well. Stateful filtering, already implemented in IPv4, is still missing in IPv6, but is on the to-do list of the Netfilter project. In commercial firewalls, IPv6 support is still very rare, but most IPv6-enabled hardware routers can already be used for filtering using their Access Control Lists.

ENCRYPTION AND AUTHENTICATION (IPSEC)

Encryption and authentication of IPv6 packets is defined in the IPsec standard, just as for IPv4. For IPv4, the FreeS/WAN project (http://www.freeswan.org/) has created patches and a related toolset for the kernel. In July 2001 the USAGI project took an IPv6 port of FreeS/WAN originally developed by the IABG project and is working on migrating it into its kernel extensions.

TEST YOUR SETUP

The classic *ping* program doesn't work with IPv6, but a new version, *ping6*, does. As root, *ping6* yourself:

```
# ping6 -n -c 1 ::1
```

You can *ping6* other link-local IPv6 addresses, but to do so you must specify the interface. The interface name can be

See IPv6, *pg. 60*

open for business.

Go with the company that offers the most comprehensive end-to-end Linux-based solutions.

Whether it's systems, workstations, business PCs, storage, software, embedded technologies, support & services, training and peripherals solutions, HP believes in giving you the choices you need to be up and running now and stay that way.

solutions that put Linux to work for you!

hp invent

call 1-888-HPLINUX today or visit www.hp.com/linux

Making the Transition to Zsh

By John Beppu

Showing someone *zsh* for the first time can be a fun experience because it's a shell with many tricks. Usually, all it takes is a short demonstration of *zsh*'s tab-completion powers to captivate your audience. Being able to type `gcc -[TAB]` to see a list of its command line options is something that most users could never imagine, but *zsh* is full of surprises when it comes to making interactive shells as functional as possible.

If you don't have *zsh* on your system, you can download it from http://www.zsh.org/, or with your distribution's automated installation tool.

However, despite how good *zsh* is, it's not widely used. The main reason for this is that it's rarely the default shell for Linux distributions. That distinction goes to GNU/bash, so it's not surprising that the majority of Linux users are *bash* users.

The main problem with *zsh* is figuring out where to begin. Its man page is located at http://zsh.sunsite.dk/Doc/ and there's a User's Guide at http://zsh.sunsite.dk/Guide/, but they are a bit overwhelming, and it's not entirely obvious how to enable some of *zsh*'s more interesting features. Fortunately, both *bash* and *zsh* implement a superset of the Bourne shell (*sh*), so porting your *bash* configuration to *zsh* is not too hard. With that in mind, let's take a look at making the transition from *bash* to *zsh* 4.0.0 or later. Along the way, we'll point out the parts of *zsh* that set it apart from other shells.

Enabling the Tab Completion System

Zsh has amazing tab-completion abilities, but people wrongly assume that it takes a long time to configure and is probably not worth the effort. Believe it or not, it only takes two lines in your `.zshrc` to enable all of *zsh*'s standard library of completions:

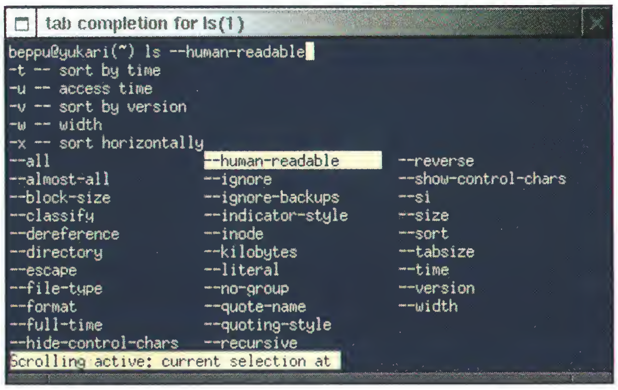
```
autoload -U compinit
compinit
```

The `compinit` function initializes the entire tab-completion system by defining a shell function for every Linux utility that *zsh* is able to tab-complete. But since it's unlikely that we'd use the tab-completion feature of every possible command in a single session, we'd like to improve the startup time of *zsh* and reduce its memory usage.

To do this, `compinit` declares every tab-completion function to be "autoloaded" (by using the word `autoload`). `autoload` tells *zsh* that a particular function exists, but that it can defer reading in the definition of the function until it's actually used. (The function that expands the tab key for a specific Linux command can be found in a file whose name contains the name of the command in a directory specified by the `FPATH` environment variable.)

The reason `compinit` is itself the target of an `autoload` command has to do with the `-U` flag. It specifies that shell aliases are not to be used when the specified function (in this case `compinit`) is run. There could be problems if a shell function relies upon an alias. In fact, the `compinit` function always uses `autoload -U` when autoloading the tab-completion functions.

FIGURE ONE: Zsh's Tab-Completion in Action



```
beppu@yukari(~) ls --human-readable
-t -- sort by time
-u -- access time
-v -- sort by version
-w -- width
-x -- sort horizontally
--all                    --human-readable       --reverse
--almost-all             --ignore               --show-control-chars
--block-size             --ignore-backups       --si
--classify               --indicator-style      --size
--dereference            --inode                --sort
--directory              --kilobytes            --tabsize
--escape                 --literal              --time
--file-type              --no-group             --version
--format                 --quote-name           --width
--full-time              --quoting-style
--hide-control-chars     --recursive
Scrolling active: current selection at
```

The end result of all this is that your `[TAB]` key now has an answer for everything. This has to be seen to be fully appreciated, so once you've added those two lines to your *.zshrc*, run *zsh* and try some of the following:

- `ls -[TAB]` will give you a list of GNU/ls options (see *Figure One*).
- `man str[TAB]` will give you a list of all man pages beginning with "str".
- `mutt -f=[TAB]` will look in your `~/Mail` directory and give you a list of mail folders to choose from.
- `w3m [TAB]` will look in your `~/.w3m/history` file and give you a list of recently visited URLs

To further customize the completion system, you can also run the `compinstall` function from the command-line. It provides a menu-driven interface that gives a good overview of some of the more exotic capabilities of the completion system. You can enable options like spell-checking, case-insensitivity, and can even add color to completion lists (similar to the way *ls* can color its output). Just pick the functionalities you want and `compinstall` will turn your choices into *zsh* commands and append them to your *.zshrc* file. It can even make a backup of the original *.zshrc* file.

More Options

But there's more to *zsh* than its completion system. There's over 150 options that can be enabled via the `setopt` command and are documented in the `zshoptions` man page.

For example, the following enables a set of options that makes directory navigation easier.

```
setopt  autocd autopushd \
        pushdignoredups
```

The `autocd` option lets you type the name of a directory to change into it: there's no need to type `cd` first anymore. Then, every time you change directories, the `autopushd` option puts that directory on the directory stack. Finally, the `pushdignoredups` prevents duplicate entries in the directory stack.

As a result, after moving around directories for a while, you can view your directory stack by typing `dirs -v`, which will yield a list of all the directories you've been in recently. See *Figure Two*.

Then, if you wanted to go back to `/usr/bin`, all you'd need to type is `~4`, and you're there.

FIGURE TWO: Examining Your Directory Stack

```
$ dirs -v
0     /var/log
1     /opt/zsh-4.0.4/share/zsh/4.0.4/functions
2     /opt/zsh-4.0.4/share/zsh/4.0.4
3     /etc
4     /usr/bin
```

There are other options in *zsh* that cater to experienced users of other shells. Some options for *csh* have funny names like `cshjunkieloops` which lets you write *csh*-style loops in *zsh*. There's also a handful of options for *sh* and *ksh* features (although only *csh* users are considered to be "junkies").

The Prompt

The prompt is another popular area of customization. Unfortunately, the `$PS1` variable from *bash* will not work "as is" in *zsh*, because *zsh* uses "`%`" as an escape character rather than "\". If you have a complex prompt in *bash*, you'll need to redo it in *zsh*. The man page to look in is `zshmisc`, and the section titled "Prompt Expansion" will have all the info you need about the "`%`" escapes.

***See Power Tools,** pg. 61*

CONFIGURATION FILES

When *zsh* starts up, there are a number of configuration files it reads from. This set of files will depend on what type of shell is being started up. A smart *zsh* user will make note of this and place configuration commands in the appropriate files.

There are two main types of shells, non-interactive and interactive. A non-interactive shell reads its commands from a file, either by a call to `zsh` *file* or because of a hash-bang (`#!`) on the first line of a shell script.

An interactive shell reads its commands from a terminal (i.e., a user typing commands at a keyboard). A login shell is a special type of interactive shell: it's been called from the *login* program, and is the type of shell a user is most likely to encounter. A non-login interactive shell would occur if a user simply typed `zsh` on the command line and is seldom encountered.

Each type of shell (login, interactive, and non-interactive) uses only a specific set of configuration files. Each configuration change should thus be examined and placed in the correct file. *Table One* shows the configuration files that each type of shell uses and is listed in the order they are read and executed.

Notice that a non-interactive shell only reads in the */etc/zshenv* and *$ZDOTDIR/.zshenv* files (if the `ZDOTDIR` environment variable is not set, then `$HOME` is used instead). Thus, any configuration change needed by a shell script must be placed in one of these two files, as no other configuration files will be read. These kinds of changes tend to only be changes to environment variables such as `PATH`.

The fact that there are both ksh-style startup files (*zprofile*) and csh-style (*zlogin*) is for historical reasons. The order in which they are read relative to the *zshrc* file corresponds to the respective shells' behavior.

Remember, the *zshenv* files are loaded every time *zsh* is started up, so it's a good idea to keep them as small as possible. Like the name implies, it should mostly be used to set up the environment.

TABLE ONE: How Zsh Starts Up

FILE	LOGIN	INTERACTIVE	NON-INTERACTIVE
/etc/zshenv	x	x	x
$ZDOTDIR/.zshenv	x	x	x
/etc/zprofile	x		
$ZDOTDIR/.zprofile	x		
/etc/zshrc	x	x	
$ZDOTDIR/.zshrc	x	x	
/etc/zlogin	x		
$ZDOTDIR/.zlogin	x		

Getting Started with SNMP

By Æleen Frisch

A large portion of a system administrator's time can be consumed by network-related tasks. Installing and configuring a network can be daunting, especially if you're starting from scratch. But monitoring and managing the network on an ongoing basis can be just as daunting, especially for very large networks. Fortunately, network monitoring and management tools can make this job easier.

Let's take a look at managing the devices that use the Simple Network Management Protocol (SNMP), the network service that underlies many network management programs. We'll start off this month with a brief look at SNMP's fundamental concepts and data entities. Next month we'll configure it on a Linux system and discuss security issues that must be resolved when using SNMP.

For a more in-depth information on the SNMP protocol, see "Network Device Interrogation" in the November 2001 issue (http://www.linux-mag.com/2001-11/snmp_01.html). A good book on SNMP is *Essential SNMP* by Douglas Maurot and Kevin Schmidt (O'Reilly & Associates, 2000).

SNMP Concepts and Constructs

SNMP was designed to be a consistent interface to gather data and set parameters for various network devices, which include switches and routers, network hosts (computers) running almost any operating system, printers, and more. It's reasonably successful at doing this once it is configured and running in the right places. The hard part is getting used to its somewhat counterintuitive terminology.

SNMP has been around since the late 1980s, and there are many versions of it (including several flavors of version 2). The versions most often implemented are version 1 and version 2c. There is also a version 3 in development as of this writing. We'll be looking at version 2c, but will address version-specific issues as they come up.

Figure One shows a basic SNMP setup. In this setup, one computer is called the Network Management Station (NMS). It's job is to collect and act on information from the various devices being monitored and/or controlled. In our example this includes two computers, a router, a network printer, and an environmental monitoring device.

In the simplest cases, the NMS periodically polls the devices it's managing, sending queries for their current status. The monitored devices respond by transmitting the requested data and can also send traps (also known as notifications in SNMP version 2). A trap is an unsolicited message to the NMS, generated when a monitored parameter reaches unacceptable levels. For example, an environmental monitoring device may send a trap when the temperature or humidity are too low or too high.

Traps are useful because they provide an asynchronous way for a device to signal that something unexpected has occurred. If an important component such as a router has a problem, you wouldn't want to wait until the next time the device was polled to find out that there is a problem.

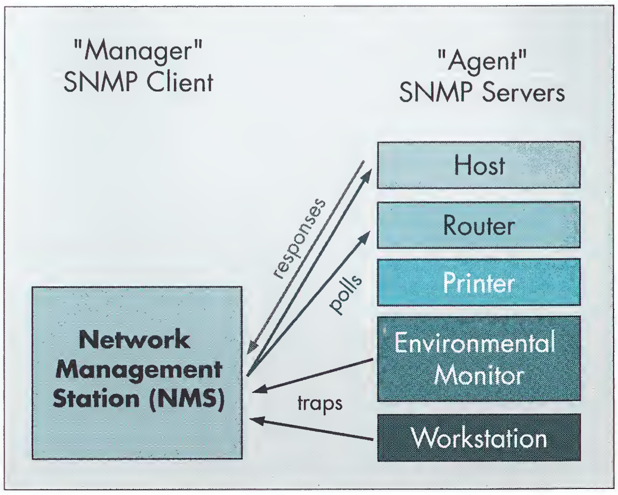
In SNMP, the term "manager" refers both to the monitoring software running on the NMS and the actual device running the software. Similarly, the term "agent" refers to the software used by the monitored devices to generate and transmit their status data as well as the device being monitored.

SNMP is a client-server protocol, but its use of "client" and "server" are reversed from typical usage: the local manager functions as the client and the remote agents function as servers. SNMP uses terminology in the same way the X Window System does: X clients run on a local host and talk to server applications on remote hosts. SNMP normally communicates on TCP and UDP ports 161, and traps use TCP and UDP ports 162, but some vendors use different ports for traps (e.g., Cisco uses TCP and UDP ports 1993).

Climbing the MIB Tree

For an SNMP manager to communicate with an agent, it must know what various data values a particular agent is monitoring. These data values and the contents associated with them are defined in Management Information Bases (MIBs). A MIB is a collection of data values and their defi-

FIGURE ONE: SNMP Manager and Agents



nitions arranged in a tree-based hierarchy.

The MIB is not a database; it doesn't actually hold any data itself. It's simply a definition of the data values that can be monitored or modified. These data definitions and naming conventions are used by the SNMP agent software, which collects the actual values for these definitions from the device and either makes them available for the NMS to query, or sends a trap if conditions warrant.

MIBs are often stored as text files for use by SNMP managers. MIBs can be standard (and are implemented by every agent), or proprietary, defining data values that are specific to a manufacturer and/or class of device.

Let's look at a sample from an actual MIB. *Figure Two* contains a subset of the overall SNMP hierarchy. The red items are leaf nodes (also known as "data nodes"), which are the only nodes in the tree that contain data definitions and values. The three data nodes (left to right) correspond to the system location description, the current number of processes on the system, and the system load average. The names of the data nodes are case-sensitive.

To access each leaf node, you'd normally start at the top (root) of the tree and work downwards, using a period to separate each data node. Thus, the data node that contains the system location description can be accessed with the string `iso.org.dod.internet.mgmt.mib-2.system.sysLocation`. This data node is defined to hold a character string, and a possible value for this data node might be "Dabney Building, 2nd Floor."

The tree structure groups data values that are related to each other. For example, although it's not illustrated here, there are various items below the `system` data node that relate to overall system (or device), including its name, physical location (`sysLocation`), and primary contact person.

It's important to note that some data nodes are considered "static," such as `sysLocation`. Once set, static data nodes are unlikely to change and would probably only be queried when the manager software starts up. Other data nodes are considered "dynamic," such as the load averages stored in the `laLoad` data node. The agent software will frequently update the data node and the manager software will query the agent for this value frequently as well.

Not every data node represented in a MIB will be relevant

Traps provide an asynchronous way for a device to signal that something unexpected has occurred.

to every device. The agent software on the device can choose what data nodes are important to it.

The top four levels of the tree (`iso.org.dod.internet`) only exist for historical reasons, and most data nodes are defined to be within the `mgmt.mib-2` and `private.enterprises` subtrees.

Although only two of `mgmt.mib-2`'s direct children are shown in *Figure Two* (`system`, which holds general information about the device, and `host`, which holds data related to computers), other important children of `mib-2` are `interfaces` (which are defined to hold information about the device's network interfaces), `ip`, `tcp` and `udp` (protocol-specific data), and `snmp` (SNMP traffic data).

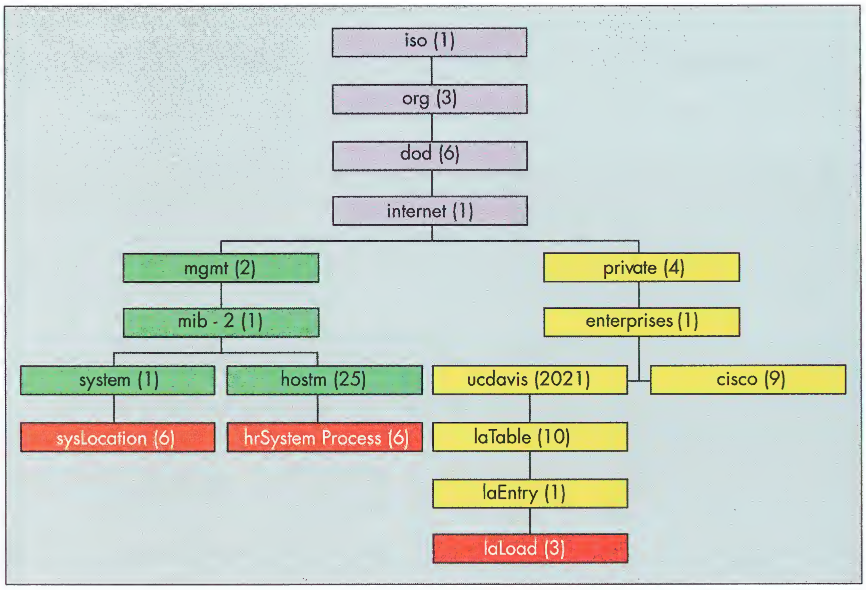
The `private.enterprises` section of the MIB tree contains vendor-specific data definitions. Every vendor is assigned a unique identifier under this node; in *Figure Two*, the nodes for the University of California-Davis and Cisco Systems are shown. For a list of all assigned numbers (and their corresponding names), see ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers. You can request a number for your organization from the Internet Assigned Numbers Authority (http://www.iana.org/cgi-bin/enterprise.pl).

The `ucdavis` subtree is important for Linux systems because the open source SNMP package generally used on Linux, Net-SNMP, has been developed by UC-Davis for a long time, and is the subtree that contains the definitions that specifically applies to Linux SNMP agents.

Each data node within the MIB

See Guru Guidance, *pg. 61*

FIGURE TWO: General SNMP MIB Hierarchy



Advanced Message Passing

By Forrest Hoffman

In the past two columns, we've looked at the theory and basics of parallel computing. This month we get to the real world with a program that performs matrix multiplication.

Parallel programming applications must make efficient use of memory and minimize inter-process communication. This requires well-conceived data structures and careful consideration of message passing strategies. In the Open Source world, you basically have two choices for your superstructure: PVM (Parallel Virtual Machine, http://www.epm.ornl.gov/pvm/) and MPI (Message Passing Interface, http://www.lam-mpi.org/ and http://www-unix.mcs.anl.gov/mpi/mpich/).

Message Passing

Point-to-point communication (message passing between individual processes) uses the `MPI_Send()` and `MPI_Recv()` routines or the `pvm_send()` and `pvm_recv()` routines. These come in blocking and non-blocking varieties.

An alternative approach is to use simultaneous message passing involving some or all processes. This method, called collective communication, uses routines called `MPI_Bcast()` and `MPI_Reduce()` for MPI and `pvm_mcast()` for PVM. The `MPI_Bcast()` and `pvm_mcast()` routines distribute the same data to all processes while `MPI_Reduce()` is used to perform some mathematical operation on a variable spread across all processes. This month we'll look at how MPI handles these tasks.

`MPI_Reduce()` has pre-defined operations to compute the sum or product of a variable, find the maximum or minimum value of a variable, or perform logical or bit-wise AND, OR, or XOR operations; however, a user-defined operation may also be performed. Collective communications are blocking, but the specification strongly suggests that they should not be relied upon for program synchronization.

A Real-World Example

Many scientific computational codes use matrix multiplication to solve a problem. Although parallel versions of subroutines that provide this functionality are easily available, you can learn a great deal about message passing strategies by writing these types of routines from scratch.

Given two matrices, **A** (an m x n matrix) and **B** (an r x p matrix), their product, **AB** = **C**, is defined only when $r = n$. That is, the number of rows of matrix **B** must equal the number of columns of matrix **A**. The product, **C**, is an m x p matrix. Each element in **C** is the inner product (or dot product) of a row vector of **A** with a column vector of **B**.

When multiplying matrices with pencil and paper, you multiply the elements in the first row of **A** with the elements in the first column of **B** and add together these products to obtain the first element of **C**. This is denoted by the blue boxes in *Figure One*. The remaining elements of **C** are calculated in similar fashion using rows of **A** and columns of **B**.

Two Algorithms

You might want to have each process calculate one or more elements of **C** by passing each both a given row of **A** and a given column of **B**. Then each element could be returned to the root process and assembled into the final matrix **C**. This strategy will work, but it requires sending the same rows or columns to multiple processes, and that requires the root process to keep track of which elements of **C** will be returned from each process.

FIGURE TWO: Compiling and running mmult.c

```
[forrest@beowulf mmult]$ mpicc -O -o mmult mmult.c
[forrest@beowulf mmult]$ mpirun -np 3 mmult
a =
 6.811383 30.782171 20.334776
 4.417596 34.088512 26.125584
b =
 9.833102 36.618156 18.256323 1.296968
 25.917124 20.557819 41.508060 35.924686
 41.968475 7.957412 24.677752 37.039406
a * b =
 1718.181885 1044.046753 1903.875610 1867.861938
 2023.365723 1070.441650 2140.317627 2198.024658
```

FIGURE ONE: Matrix Multiplication

AB=C

$$\underbrace{\begin{bmatrix} \boxed{a_{11}} & a_{12} & a_{13} \\ \boxed{a_{21}} & a_{22} & a_{23} \end{bmatrix}}_{m \times n} \underbrace{\begin{bmatrix} \boxed{b_{11} & b_{12} & b_{13} & b_{14}} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{bmatrix}}_{n \times p} = \underbrace{\begin{bmatrix} \boxed{a_{11}b_{11}} + a_{12}b_{21} + a_{13}b_{31} & \boxed{a_{11}b_{12}} + a_{12}b_{22} + a_{13}b_{32} & \boxed{a_{11}b_{13}} + a_{12}b_{23} + a_{13}b_{33} & \boxed{a_{11}b_{14}} + a_{12}b_{24} + a_{13}b_{34} \\ \boxed{a_{21}b_{11}} + a_{22}b_{21} + a_{23}b_{31} & \boxed{a_{21}b_{12}} + a_{22}b_{22} + a_{23}b_{32} & \boxed{a_{21}b_{13}} + a_{22}b_{23} + a_{23}b_{33} & \boxed{a_{21}b_{14}} + a_{22}b_{24} + a_{23}b_{34} \end{bmatrix}}_{m \times p}$$

An alternative strategy would be to send a column of **A** and a row of **B** to each process. In this case each process would calculate one of the products contained in each element of **C**. These products could be summed by a global reduction routine to produce the final result. (See the red boxes in *Figure One*.) This strategy is easier to program and minimizes communications prior to the calculations, but it requires more communications afterward; it thus relies upon an efficient implementation of the reduction algorithm in the message passing system. Most MPI implementations use a tree-based algorithm to collate the data so that the reduction operations are actually distributed when more than four processes are involved in the operation.

While `MPI_Bcast()` is useful for distributing parameters or data common to calculations being performed by all processes, it is not useful for distributing subsets of data in a regular fashion to all processes. This is accomplished in MPI using the `MPI_Scatter()` routine. This routine effectively subsets the data and sends individual subsets to each process in turn according to its rank. A variant of this routine, called `MPI_Scatterv()`, can be used to send a varying amount of data to each process. Similarly, the `MPI_Gather()` and `MPI_Gatherv()` routines are used to collect subsets of data from processes and assemble them into a larger array.

Listing One contains the code for this alternative matrix multiplication strategy. To keep the code small for this column, the matrix sizes — `M`, `N`, and `P` — are pre-defined, the code is all in `main()`, the data are read from files, and the program requires `N` processes to be used in the calculation. This code can easily be converted into a subroutine that can handle any size matrix and utilize any number of processes less than or equal to `N`.

***See Extreme Linux,** pg. 62*

LISTING ONE: mmult.c

```c
#include <stdio.h>
#include <unistd.h>
#include "mpi.h"

#define M  2
#define N  3
#define P  4

int main(int argc, char **argv)
{
  FILE *fpa, *fpb;
  int me, nprocs, namelen, i, j;
  float a[M][N], b[N][P], c[M][P], d[M][P];
  float atrans[N][M], avec[M], bvec[P];
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  MPI_Status status;

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &me);
  MPI_Get_processor_name(processor_name, &namelen);

  if (!me) {
    if (!(fpa=fopen("a.dat", "r")) ||
        !(fpb=fopen("b.dat", "r")))
      exit(-1);
    printf("a =\n");
    for (i = 0; i < M; i++) {
      for (j = 0; j < N; j++) {
        fscanf(fpa, " %f ", &a[i][j]);
        atrans[j][i] = a[i][j];
        printf(" %f", a[i][j]);
      }
      printf("\n");
    }
    printf("b =\n");
    for (i = 0; i < N; i++) {
      for (j = 0; j < P; j++) {
        fscanf(fpb, " %f ", &b[i][j]);
        printf(" %f", b[i][j]);
      }
      printf("\n");
    }
    fclose(fpa);
    fclose(fpb);
  }

  if (MPI_Scatter(atrans, M, MPI_FLOAT, avec, M,
    MPI_FLOAT, 0, MPI_COMM_WORLD) != MPI_SUCCESS)
    fprintf(stderr, "Error scattering A\n");

  if (MPI_Scatter(b, P, MPI_FLOAT, bvec, P,
    MPI_FLOAT, 0, MPI_COMM_WORLD) != MPI_SUCCESS)
    fprintf(stderr, "Error scattering B\n");

  for (i = 0; i < M; i++)
    for (j = 0; j < P; j++)
      c[i][j] = avec[i] * bvec[j];

  if (MPI_Allreduce(c, d, (M*P), MPI_FLOAT,
    MPI_SUM, MPI_COMM_WORLD) != MPI_SUCCESS)
    fprintf(stderr, "Error reducing\n");

  if (!me) {
    printf("a * b =\n");
    for (i = 0; i < M; i++) {
      for (j = 0; j < P; j++)
        printf(" %f", d[i][j]);
      printf("\n");
    }
  }

  MPI_Finalize();
}
```

Linux and Executables

By Benjamin Chelf

For the past few months, we've been learning how the compiler and linker work together to take the programs you write and convert them into executables that the operating system can run. We've followed the process from source code to object module to executable, with static and shared libraries thrown in as well.

But when it's time to actually run your executable, what does Linux do for you?

What The Shell Does

Let's say you've just finished compiling a program to run. Use the code found in *Listing One* and *Listing Two* as our examples. It's a simple executable that loads a shared library and calls a function in that shared library. It then calls the `sleep()` system call to suspend itself for a while (so we can find out information about it while it's running) before exiting. Compile the shared library by running:

```
machine:~/> gcc -fPIC -c b.c
machine:~/> ld -shared -o b.so b.o
```

Then compile the program with:

```
machine:~/> gcc main.c -ldl
```

This generates an executable file named *a.out*. So what exactly happens when you run *a.out* at your prompt?

```
machine:~/> ./a.out
```

After checking for shell functions or aliases (and finding none for this command), the shell calls `fork()` so that now two copies of the shell are running (for more on the `fork()` system call, see the Spring 1999 Compile Time column online at http://www.linux-mag.com/1999-05/compile_01.html). After the `fork()` returns, the resulting child shell process calls the `exec()` system call, which causes the operating system to load and run the *a.out* program in place of the child shell, effectively killing the shell (but not the process). The parent shell process waits for *a.out* to finish, then issues another prompt.

The `exec()` system call (short for execute), is one of many different system calls that control program execution. Each one works a little bit differently, so the calling program needs to carefully choose which one to use. The prototypes for the different versions of `exec()` are listed in *Figure One*.

LISTING ONE: main.c — A Program That Uses a Shared Library

```
#include <stdio.h>
#include <dlfcn.h>

int main ()
{
  void* handle;
  void (*printer)(void);
  char* error;

  handle = dlopen ("/home/chelf/linuxmag/0202/b.so", RTLD_LAZY);

  error = dlerror ();
  if (error)
    {
      printf ("%s\n", error);
      exit (1);
    }

  printer = dlsym (handle, "b_printer");

  error = dlerror ();
  if (error)
    {
      printf ("%s\n", error);
      exit (1);
    }

  printer ();

  sleep (200);

  dlclose (handle);
}
```

LISTING TWO: b.c — A Shared Library Source Module

```
#include <stdio.h>

void b_printer ()
{
  printf ("Printing something from a shared library.\n");
}
```

FIGURE ONE: Prototypes for the 'exec' system call found in <unistd.h>

```
int execl  (const char *path, const char *arg, ...);
int execlp (const char *file, const char *arg, ...);
int execle (const char *path, const char *arg , ..., char* const envp[]);
int execv  (const char *path, char *const argv[]);
int execvp (const char *file, char *const argv[]);
int execve (const char *filename, char *const argv [], char *const envp[]);
```

The functions that end in a "p" (`execlp()` and `execvp()`) will search your path for the given `file`. For example, if the program calls `execlp ("gcc", ...)`, the system will look for the file "gcc". For the functions that do not end in a "p", you must specify the entire path to the program, including the filename, as the `path` argument.

Functions that have an "l" in their name (`execl()`, `execlp()`, and `execle()`) have the program's arguments passed as arguments to the specific `exec` function. There can be as many arguments as you like.

Functions that have a "v" in their name (`execv()`, `execvp()`, and `execve()`) have the program's arguments passed via a single pointer to an `argv`-like structure (a pointer to an array of `char *`'s).

Functions that end in an "e" (`execle()` and `execve()`) allow you to pass an environment to the executable in addition to the program's arguments. This lets you adjust environment variables like `HOME` or `PATH`, or add or remove any environment variables you want. The environment is always passed via a pointer to an array of `char *`'s (just as the arguments are passed in the "v" functions).

What the Kernel Does

Once the child shell calls `exec()` to start running *a.out*, the kernel has to do some work before before the program can begin execution. Let's take a closer look at the *a.out* program.

Figure Two shows an edited sample of the output generated by running `objdump -s a.out`. After printing the type of executable *a.out* is, it lists its various sections (that's what the `-s` flag controls).

The exact contents within each section aren't that important; we're interested in the section names right now. The sections that the kernel cares most about are `.text`, `.data`, and `.bss`. In *FigureTwo*, *objdump* shows the `.bss` section as `.sbss` which can be considered the same as the `.bss` section. More on what these sections contain in a bit.

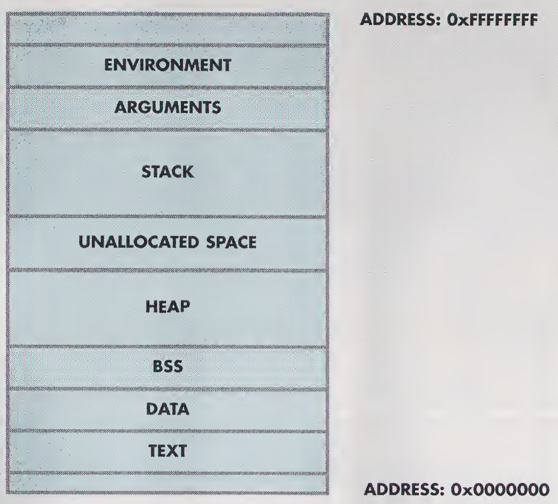
In the March issue (http://www.linux-mag.com/2002-03/compile_01.html), we showed how a process virtual address space is divided into various pieces, more properly called "segments." *Figure Three* (pg. 46) shows a more detailed map of the virtual address space.

Below the heap, there are three separate segments (text, data, and bss) that match the sections in *a.out* that *objdump* reported, which, as you might have guessed, isn't a coincidence.

FIGURE TWO: Output of the command objdump -s a.out

```
a.out:     file format elf32-i386

Contents of section .text:
 80484e0 31ed5e89 e183e4f8 50545268 dc860408  1.^.....PTRh....
 80484f0 68048404 08515668 dc850408 e87bffff  h....QVh.....{..
 8048500 fff49090 5589e553 50e80000 00005b81  ....U..SP.....[.
 8048510 c3661200 008b8338 00000085 c07402ff  .f.....8.....t..
 8048520 d08b5dfc c9c39090 90909090 90909090  ..].............
 8048530 558b155c 97040889 e583ec08 85d27549  U..\..........uI
 8048540 8b155897 04088b02 85c0741a 8d742600  ..X.......t..t&.
 8048550 8d4204a3 58970408 ff128b15 58970408  .B..X.......X...
 8048560 8b0a85c9 75eab84c 84040885 c0741083  ....u..L.....t..
 8048570 ec0c6860 970408e8 d0feffff 83c410b8  ..h`............
 8048580 01000000 a35c9704 0889ec5d c38d7600  .....\.....]..v.
 8048590 5589e583 ec0889ec 5dc38db6 00000000  U.......].......
 80485a0 5589e5b8 2c840408 83ec0885 c0741583  U...,........t..
 80485b0 ec086858 98040868 60970408 e86bfeff  ..hX...h`....k..
 80485c0 ff83c410 89ec5dc3 908db426 00000000  ......]....&....
 80485d0 5589e583 ec0889ec 5dc39090 5589e583  U.......]...U...
 80485e0 ec1883ec 086a0168 20870408 e8dbfeff  .....j.h .......
 80485f0 ff83c410 89c08945 fce86efe ffff89c0  .......E..n.....
 8048600 8945f483 7df40074 1f83ec08 ff75f468  .E..}..t.....u.h
 8048610 3f870408 e873feff ff83c410 83ec0c6a  ?....s.........j
 8048620 01e896fe ffff89f6 83ec0868 43870408  ...........hC...
 8048630 ff75fce8 04feffff 83c41089 45f8e829  .u..........E..)
 8048640 feffff89 c08945f4 837df400 741e83ec  ......E..}..t...
 8048650 08ff75f4 683f8704 08e82efe ffff83c4  ..u.h?..........
 8048660 1083ec0c 6a01e851 feffff90 8b45f8ff  ....j..Q.....E..
 8048670 d083ec0c 68c80000 00e8defd ffff83c4  ....h...........
 8048680 1083ec0c ff75fce8 20feffff 83c410c9  .....u.. .......
 8048690 c3909090 90909090 90909090 90909090  ................
 80486a0 55a16497 040889e5 5383ec04 83f8ffbb  U.d.....S.......
 80486b0 64970408 74168d76 008dbc27 00000000  d...t..v...'....
 80486c0 83eb04ff d08b0383 f8ff75f4 585b5dc3  ..........u.X[].
 80486d0 5589e583 ec0889ec 5dc39090            U.......]...
Contents of section .data:
 8049750 00000000 00000000 70970408 00000000  ........p.......
Contents of section .sbss:
```

FIGURE THREE: A process's virtual address space



Copying Sections to Segments

If you were to follow the Linux kernel as it ran through one of the `exec()` functions, it would eventually lead to a function in the kernel called `load_elf_binary()` (located in the kernel source file *fs/binfmt_elf.c*) that copies the data from the `.bss` section of the executable into the bss segment in the new process's virtual address space. It also copies the `.data` section into the new data segment, and the `.text` section into the new text segment.

You may be asking what's *in* these sections and/or segments? The text segment contains the instructions to be executed, the data segment contains the static and global data that is initialized, and the bss segment contains global data that is uninitialized. "Global data" corresponds to variables or constants declared outside the scope of any function, such as the `main()` function. "Static data" corresponds to variables or constants that have been declared as `static`, even those inside functions. The compiler will automatically initialize all variables, so the bss segment is often empty.

After the segments are set up and the arguments to the program and its environment are copied into the address space, `load_elf_binary()` calls the `start_thread()` function, and the program begins executing the code in its text segment.

What About Shared Libraries?

Last month, we explained how you can dynamically load object files into your programs. But how does this work once the operating system has already set up the processes address space as described above? Each new library that is dynamically loaded gets its own text, bss, and data segments in the process address space. The operating system stacks them up as they are loaded. You can see this best by looking at a process while it's running.

You can get a map of a running process's virtual address space by looking in the file */proc/*`pid`*/maps*, where `pid` is the process ID of the one we're interested in. To get the process ID, use the *ps* command:

```
machine:~/> ps ax | grep a.out
23098 pts/5    S      0:00 ./a.out
machine:~/>
```

To see the process's virtual address map, run:

```
machine:~/> more /proc/23098/maps
```

FIGURE FOUR: A map of the address space of a.out

```
08048000-08049000 r-xp 00000000 /home/chelf/linuxmag/0202/a.out
08049000-0804a000 rw-p 00000000 /home/chelf/linuxmag/0202/a.out
40000000-40015000 r-xp 00000000 /lib/ld-2.2.4.so
40015000-40016000 rw-p 00014000 /lib/ld-2.2.4.so
40016000-40017000 rw-p 00000000
40017000-4001b000 r-xp 00000000 /lib/libsafe.so.1.3
4001b000-4001c000 rw-p 00003000 /lib/libsafe.so.1.3
4001c000-4001d000 r-xp 00000000 /home/chelf/linuxmag/0202/b.so
4001d000-4001e000 rw-p 00000000 /home/chelf/linuxmag/0202/b.so
4001e000-4001f000 rw-p 00000000
40025000-40027000 r-xp 00000000 /lib/libdl-2.2.4.so
40027000-40029000 rw-p 00001000 /lib/libdl-2.2.4.so
40029000-40155000 r-xp 00000000 /lib/libc-2.2.4.so
40155000-4015a000 rw-p 0012b000 /lib/libc-2.2.4.so
4015a000-4015f000 rw-p 00000000
bfffe000-c0000000 rwxp fffff000
```

Figure Four shows the (edited) map of the process space of the executable (this is why we added the call to `sleep()` in *Listing One*). The numbers in the first column show the virtual address ranges for each of the segments. The first two are for the text and data segments of *a.out*. There's no bss segment in the process map since there was no data in the bss section of *a.out*.

You can identify the text segment by the "x" in the second column (it stands for "executable," just like in the output from *ls -l*). The "w" means the segment is writable and identifies the data segment. Executable segments cannot be written on because Linux doesn't allow self-modifying code. All segments must be readable (the "r" in the second column).

Below the segments for our executable are the segments for the "ld" library. In this case, "ld" doesn't stand for the *ld* linker, but rather what's called the program interpreter (yes, that's a very poor name for it). The program interpreter analyzes the executable, figures out which shared libraries are necessary for it to run, and locates and loads them into the virtual address space for the executable. In this case, it brought in *libc*, *libdl*, and *libsafe*.

The *libsafe* library (along with *libc*) is always linked into executables by the *ld* linker. It ensures that you don't overflow your stack or try to write past what you are currently allowed. (For more on libsafe, see http://www.gnu.org/ directory/ libsafe.html).

The sections that the kernel cares about are .text, .data, and .bss.

Also in the map, we can see the shared library we built, *b.so*. However, this library wasn't loaded into the virtual address space by the program interpreter. That's because *b.so* wasn't specified on the command line that built *a.out*. Instead, it was the *dl* library (and the `dlopen()` function) that loaded *b.so*.

In a sense, the *dl* library duplicates some of what the *ld* program interpreter does. The difference is *when* these two libraries load other shared libraries into the process's virtual address space: the program interpreter does it when the process starts up, the *dl* library does it any time after the process has begun executing.

But That's Not All!

Believe it or not, there's still a lot that hasn't been covered. The Linux kernel (in those `exec()` functions) does a lot of other things to make your executables run. You can consult the kernel source file *fs/exec.c* or the last chapter of the O'Reilly book *Understanding the Linux Kernel* if you'd like to learn more.

An interesting thing you'll find if you look in the source is that the Linux kernel is not limited to running ELF executables. You can also run executables of the old a.out format (not to be confused with the program in our example) on your system.

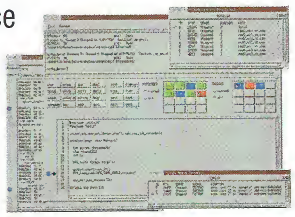
So hopefully you can now see that although it might not seem like it, Linux is hard at work "under the hood" to make your programs run.

In the meantime, happy hacking!

Benjamin Chelf is an author and engineer at CodeSourcery, LLC. He can be reached at chelf@codesourcery.com.

DEBUG & PROFILE Linux Clusters

Now debug & profile MPI and OpenMP parallel applications on Linux Clusters. The Portland Group™ Compiler Technology CDK™ v4.0 package, with the new MPI Cluster PGDBG® Debugger and PGPROF® Profiler, offers turn-key installation to quickly transform your Linux Cluster into a productive, effective parallel programming system. CDK tools include native high performance OpenMP & auto-parallel F90, F77, C and C++ compilers; HPF compiler; MPI communication library; scalar and parallel math libraries; parallel batch management queueing tools; and extensive training material. Premium compilers and tools for professional developers. It's the best investment you can make in your Linux cluster.



The Portland Group™ Compiler Technology
STMicroelectronics
++01 (503) 682-2806 www.pgroup.com sales@pgroup.com

Setting Up PostNuke

By Jeremy Zawodny

Last month began our look at a class of Web-based tools for creating dynamic Web sites. The LAMP-powered tools provide a framework for news and announcements, threaded discussions, weblogs, polls, and dynamic content. This month we'll go through the process of installing and setting up PostNuke, a popular PHP-based system.

Before we get too far, it's worth mentioning that you should try out PostNuke on a test server or at least a new virtual server so as not to disrupt your normal Web site operations.

Getting the Code

To get started, in addition to the normal Apache and MySQL setup, make sure you have at least version 4.0.1 pl2 of PHP installed. Then head over to the PostNuke Web site (http://www.postnuke.com/) to download the latest stable version. At the time of this writing it is .703, which comes packaged in a file named `PN_703.tar.gz`.

Untarring the file with the command `tar -zxvf PN_703.tar.gz` creates the directory `pn703` that contains a *README* file as well as two subdirectories, *sql* and *html*. We'll use the files in the *html* directory after preparing the database.

Database Setup

The easiest way to set up MySQL for PostNuke is to create a separate database for the PostNuke site. We'll create one called *nuke:*

```
$ mysqladmin -u root -p create nuke
```

Then we'll create a username and password that will have full access to the nuke database (but nothing else). The command sequence is in *Figure One*.

If we don't create the database in advance, the PostNuke installation script would need to use a MySQL account (such as `root`) that had privileges to create databases. For security reasons, it's generally not a good idea to have applications running with administrator access to MySQL.

Post Nuke Setup

With the database work out of the way, it's time to get the PHP and HTML files in order. It's probably best to create a new *nuke* directory underneath the document root. If that's located at */www*, then you'd use the commands:

```
# mkdir /www/nuke
# cp -r html/* /www/nuke/
```

Then point your browser at the `install.php` URL (http://localhost/nuke/install.php) to begin the configuration process. The installation routine walks you through several pages and collects information about your setup.

The first page displays the GPL license and waits for you to click the *Next* button to continue with the installation process.

The second page will ask you to check the permissions of the PostNuke configuration file, saying something like:

> We will first check to see that your CHMOD settings are correct in order for the script to write to the file. If your settings are not correct, this script will not be able to encrypt your data in your config file. Encrypting the SQL data is added security, and is set by this script. You will also not be able to update your preferences from your admin once your site is up and running. Please set your CHMOD on config.php to 666 so this script can write and encrypt the DB data.

While this probably sounds like a Really Bad Idea, the Web server needs to be able to write to *config.php* and *config-old.php* to save your settings, and it'll only be for a short while. So change the file permissions for now:

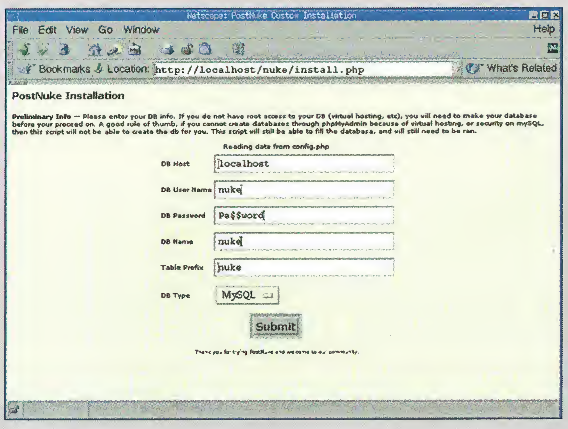
```
# chmod 666 /www/nuke/config.php
# chmod 666 /www/nuke/config-old.php
```

FIGURE ONE: Creating the *nuke* MySQL user

```
$ mysql -u root nuke

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 21 to server version: 3.23.47-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> GRANT ALL PRIVILEGES ON nuke.*
       TO nuke@localhost IDENTIFIED BY 'password';

Query OK, 0 rows affected (0.11 sec)
mysql> quit
```

FIGURE TWO: The PostNuke database settings

DB HOST:	localhost	**DB NAME:**	nuke
DB USER NAME:	nuke	**TABLE PREFIX:**	nuke
DB PASSWORD:	*password*	**DB TYPE:**	MySQL


Netscape: PostNuke Custom Installation

Location: http://localhost/nuke/install.php

PostNuke Installation

Preliminary Info -- Please enter your DB info. If you do not have root access to your DB (virtual hosting, etc), you will need to make your database before your proceed on. A good rule of thumb, if you cannot create databases through phpMyAdmin because of virtual hosting, or security on mySQL, then this script will not be able to create the db for you. This script will still be able to fill the database, and will still need to be ran.

Reading data from config.php

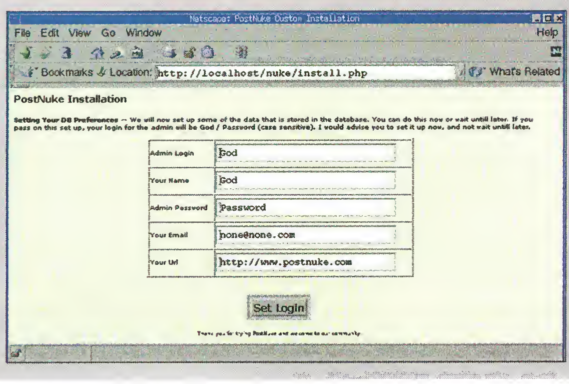
DB Host	localhost
DB User Name	nuke
DB Password	Pa$$word
DB Name	nuke
Table Prefix	nuke
DB Type	MySQL

Submit

Thank you for trying PostNuke and welcome to our community.

FIGURE THREE: General PostNuke settings

ADMIN LOGIN:	God
YOUR NAME:	God
ADMIN PASSWORD:	*Password*
YOUR EMAIL:	none@none.com
YOUR URL:	http://www.postnuke.com


Netscape: PostNuke Custom Installation

Location: http://localhost/nuke/install.php

PostNuke Installation

Setting Your DB Preferences -- We will now set up some of the data that is stored in the database. You can do this now or wait untill later. If you pass on this set up, your login for the admin will be God / Password (case sensitive). I would advise you to set it up now, and not wait untill later.

Admin Login	God
Your Name	God
Admin Password	Password
Your Email	none@none.com
Your Url	http://www.postnuke.com

Set Login

Thank you for trying PostNuke and welcome to our community.

We'll be sure to set them to something more secure as soon as the configuration is finished. Once the file permissions are changed, click the *ReCheck* button on that page. It will confirm that the files are writable and present a *Continue* button to click if everything is set up correctly.

The next page allows you to configure the database connection parameters. We'll need to verify that the settings are correct for our setup. *Figure Two* shows what the page should look like with all the correct values.

After you've entered all the information, the next page will allow you to double-check it. Once you're sure it's correct, click the *New Install* button to start installation.

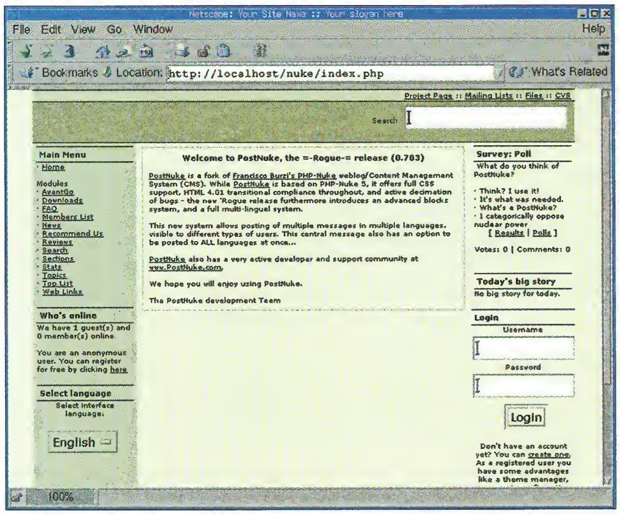
The next page confirms the information once more and then asks if you'd like to create the database. Leave that option unchecked, since we created it earlier. Click the *Start* button to continue.

The PostNuke installer will create all the necessary tables and tell you that it has done so. Click *Continue* to move on.

The next page, shown in *Figure Three*, covers some general PostNuke preferences. You should change the default administrative username ("`God`") and password ("`Password`") as well as the e-mail address and site URL to whatever is appropriate for your site.

Once that's completed, the next page will confirm that all the settings were saved properly. If they were, click the *Finish* button and you're done. The final page will display credits and a *Go to your PostNuke site* link which should take you to your new site.

FIGURE FOUR: Your New PostNuke Site


Netscape: Your Site Name :: Your slogan here

Location: http://localhost/nuke/index.php

Project Page :: Mailing Lists :: Files :: CVS

Search

Main Menu
- Home

Modules
- AvantGo
- Downloads
- FAQ
- Members List
- News
- Recommend Us
- Reviews
- Search
- Sections
- Stats
- Topics
- Top List
- Web Links

Who's online
We have 1 guest(s) and 0 member(s) online.

You are an anonymous user. You can register for free by clicking here

Select language
Select interface language:
English

Welcome to PostNuke, the =-Rogue-= release (0.703)

PostNuke is a fork of Francisco Burzi's PHP-Nuke weblog/Content Management System (CMS). While PostNuke is based on PHP-Nuke 5, it offers full CSS support, HTML 4.01 transitional compliance throughout, and active decimation of bugs - the new 'Rogue release furthermore introduces an advanced blocks system, and a full multi-lingual system.

This new system allows posting of multiple messages in multiple languages, visible to different types of users. This central message also has an option to be posted to ALL languages at once...

PostNuke also has a very active developer and support community at www.PostNuke.com.

We hope you will enjoy using PostNuke.

The PostNuke development Team

Survey: Poll
What do you think of PostNuke?
- Think? I use it!
- It's what was needed.
- What's a PostNuke?
- I categorically oppose nuclear power

[Results | Polls]

Votes: 0 | Comments: 0

Today's big story
No big story for today.

Login
Username
Password
Login

Don't have an account yet? You can create one. As a registered user you have some advantages like a theme manager,

100%

Fix Permissions

Before we forget, let's go back and fix those file permissions that we adjusted earlier. Because there is now a username and password stored there, it's best to tighten security by making sure that only the Web server can read them.

```
# chmod 600 /www/nuke/config.php
# chmod 600 /www/nuke/config-old.php
# chown www /www/nuke/config.php
# chown www /www/nuke/config-old.php
```

This doesn't prevent someone with access to your Web server from writing a CGI or PHP script to grab this configuration data, but it will at least keep out folks who are casually snooping around.

See Lamp Post, *pg. 63*

JavaServer Pages to the Rescue

By Hans Bergsten

What we expect from a Web site has changed dramatically over the last few years. In the early days of the Web, just finding a site with useful information was a thrill. Today we expect Web sites to be highly dynamic with advanced search capabilities, personalization, online ordering, and accurate shipment tracking functions — all accessible through an easy-to-use, visually appealing user interface. Developing this type of site requires the cooperation of many people with different skills. Perhaps the largest challenge is keeping the request-processing code and HTML markup separate so they can be worked on independently. JavaServer Pages (JSP) is a popular technology that can be used to accomplish this.

Why JSP?

Java has proven itself as a great language and platform for server-side applications. Java servlets first appeared in 1997 and have been embraced by all major Web servers as the alternative to Common Gateway Interface (CGI) scripts. As described in last month's column (available online at http://www.linux-mag.com/2002-04/java_01.html), servlets are applications created by extending certain Java classes. They are managed by a Web container that provides the runtime environment and access to other resources. The container runs in a permanent process, giving servlets a performance advantage over CGI scripts, which can require a new process to be created for each request.

But there's a problem with servlets: besides the request-processing code, a servlet also includes statements to emit the HTML elements for the response. This makes it virtually impossible for a Web designer without programming experience to modify the design of the Web application, since even a minor design changes requires help from a programmer.

JavaServer Pages (http://java.sun.com/products/jsp) were added to the Java toolbox in 1999 to help solve this problem. A JSP is a text file (with a *.jsp* suffix) that includes standard HTML elements along with JSP elements (that look similar to HTML) to control the dynamic portions of the page. These can be things such as search results, shopping cart contents, or a delivery tracking number. The JSP elements map to Java methods that are called when Web server processes the request for the JSP. By placing the application logic into a simple HTML-like element, anyone familiar with HTML can add dynamic behavior to a Web page without needing to know Java programming. Conversely, a programmer can develop the application in Java without needing to know how the output will be presented.

While it may sound like JSP replaces servlets, the truth is that JSP is typically used in combination with servlets. In fact, there's an even closer connection between JSP and servlets.

When the Web container receives the first request for a JSP, it actually converts the JSP into a servlet. This servlet is then compiled and executed by the container, and sends its response to the browser. Subsequent requests for the same JSP invoke the already-compiled servlet. When a JSP is modified, the next request for that page causes the container to translate and compile the modified file.

Let's take a closer look at what a JSP file might look like.

LISTING ONE: Sample JSP Page: foo.jsp

```
<%@ page contentType="text/html" errorPage="/error.jsp" %>
<%@ taglib prefix="c" uri="http://java.sun.com
  /jstl/ea/core" %>
<%@ taglib prefix="mylib" uri="http://mycompany.com
                                 /mylib" %>

<html>
  <body>
    <h1>Product List</h1>
    Here's a list of our products as of
    <%= new java.util.Date() %>

    <mylib:getProducts var="productList" />

    <table>
      <c:forEach items="$productList" var="current">
        <tr>
          <td><c:expr value="$current.name" /></td>
          <td><c:expr value="$current.descr" /></td>
          <td><c:expr value="$current.price" /></td>
        </tr>
      </c:forEach>
    </table>

    <jsp:include page="/footer.jsp" />
  </body>
</html>
```

JSP Elements

As mentioned previously, a JSP is a text file with a *.jsp* file extension to tell the server what it is. The file contains static content plus the JSP elements. The static content is called *template text* and can be anything the client can handle (HTML, WML, or text). *Listing One* shows a sample JSP file.

There are three types of JSP elements that can be used in a page: *directive elements*, *scripting elements*, and *action elements*.

DIRECTIVE ELEMENTS

A directive element has the form `<%@ directive attr="value" ... %>` and describes the page itself. These are things that are the same no matter when, or by whom, the page is requested.

There are three directive elements, on lines 1-3. The `page` directive on line 1 has two attributes that define the type of content (MIME type) the page will contain and an error page to return if there are any runtime errors.

Lines 2 and 3 contain the `taglib` directive. This directive shows that we wish to make use of a custom tag library. We'll go into more detail on custom tag libraries when we cover action elements below. For now, just notice that each use of `taglib` has a different value for its `prefix` attribute.

SCRIPTING ELEMENTS

Scripting elements were the original way to add dynamic behavior in JSP, by allowing you to place Java code in the JSP file. There are three types of scripting elements:

1. A scriptlet (enclosed within `<% ... %>`) that can be used to execute any number of Java statements.

2. An expression (enclosed within `<%= ... %>`) that executes one Java expression and places the result in the response.

3. A declaration (enclosed within `<%! ... %>`) that declares a Java variable or method.

Line 9 contains a scripting expression of Java code that creates a new `java.util.Date` object. The container executes the expression, creating the object. It then adds a string representation of the object to the response. This places the current date in the response.

You should avoid using too many scripting elements because it takes us back to the problem we set out to solve: a mixing of code and markup elements. That's why there's an easier way: action elements.

ACTION ELEMENTS

Action elements are the preferred way to add dynamic behavior to a Web page, and have mostly superseded scripting elements. JSP supports two types of actions: *standard actions* and *custom actions*. Both types must have an XML-style namespace prefix to uniquely identify them.

Standard actions use the `jsp` prefix and are defined as part of the JSP specification (http://java.sun.com/products/jsp). Line 23 contains a standard action element, `<jsp:include>`. This action functions like the C and C++ `#include` directive, placing the contents of the `page` attribute directly into the the response. In this case, since the `page` is another JSP, that page is executed and its output placed into the response. This action is typically used for shared page fragments such as headers and footers.

Other standard actions can let you access properties in a JavaBean or even let another Web resource continue the processing of a request. (For more on JavaBeans, see "Bean Soup.")

Custom actions are Java classes that follow a specific interface. These classes can access the entire set of Java APIs which let you do nearly anything you can do in Java. They are packaged into custom tag libraries that also contain a mapping of an action element to a Java method and are accessed from a JSP via the `uri` attribute of the `taglib` directive element.

Each action element also contains a `prefix` attribute. In lines 2 and 3, we've loaded two tag libraries and assigned them the prefixes `c` and `mylib`. These custom prefixes are used

BEAN SOUP

JavaBeans is the term used for a Java class that represents a "component" in its broadest sense. They are often used in JSP and servlets to encapsulate information (called properties) about data, such as customers and products. Each instance of a JavaBeans class is called a bean, and every bean contains only properties.

A JavaBeans class has a no-argument constructor. To read and set a property, you use the `getPropertyName()` and the `setPropertyName()` methods.

A typical use for a bean would hold information about a GUI widget, such as a text box. It might have properties for the text, font, color, size, etc. Another application could read and modify the bean's properties.

Beans are also often used to represent business objects, such as a customer bean with properties like name, address, and telephone number. You can think of a bean as a container of information that can be discovered at runtime and accessed by other classes in a generic way, such as by the `<c:expr>` action elements shown in *Listing One*.

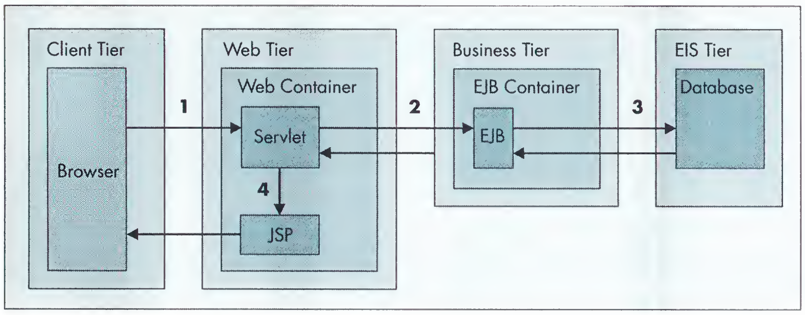
Despite the similarity in names, JavaBeans and Enterprise JavaBeans (EJB) don't have a lot in common. Although EJB's are also components similar to JavaBeans, they must follow a large set of rules and can only be used within a special EJB container that's part of the J2EE package. For more on Enterprise JavaBeans, see Sun's J2EE Web site (http://java.sun.com/j2ee/).

with the action element to specify which library contains that specific custom action.

For example, the action corresponding to the `<mylib:getProducts>` action element on line 11 can be found in the library associated with the `mylib` prefix by the `taglib` directive on line 3.

The call on line 11 to `<mylib:getProducts>` specifies that the container should call the `getProducts` action in the tag library associated with `mylib`. This calls a Java method (probably, but not necessarily called `getProducts()`). We don't know exactly what it does, but based on its name, we can assume that it retrieves product information from a database, and saves it in the variable `productList`, as specified by the `var` attribute of the `getProducts` element. Since a custom action is, as its name implies, custom-made for a specific application, it's up to the Java programmers on the team to decide which custom actions to develop, what they should do, and what they return.

FIGURE ONE: J2EE Request Flow



Custom Tag Libraries and the JSTL

Many projects have "reinvented the wheel" by writing custom actions to handle simple tasks. The Apache Jakarta Taglibs Project (http://jakarta.apache.org/taglibs) has a number of custom tag libraries that handle various tasks. One of these is the JSP Standard Tag Library (JSTL, http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html), which is working to set standards that define actions for database access, flow control, XML processing, external resource access, and internationalization, as well as a language for easy access to request parameters and other data. The JSTL is currently in its Early Access Release and is scheduled to be officially released this summer.

The `taglib` directive on line 2 loads the JSTL Early Access library, and uses "c" as its prefix. The `<c:forEach>` action on line 14 is a loop that evaluates its body (the elements from lines 15 through to the end of the `<c:foreach>` action on line 20) once for each element in the collection specified by the `items` attribute, storing the current element in the variable named by the `var` attribute. In this example, the collection contains JavaBeans representing products which were retrieved by the `<mylib:getProducts>` custom action on line 11. The three `<c:expr>` actions on lines 16, 17, and 18 put the values of the current bean's properties into the response.

JSP and J2EE

JSP is one of the technologies that makes up the J2EE, or Java 2 Enterprise Edition. J2EE is a collection of Java technologies that are useful for the server side of the enterprise application equation.

For enterprise applications, the system components are often assigned to different "tiers," that can run on the same or different servers to provide scalability. In J2EE 1.3, the four tiers are the Client Tier, the Web Tier, the Business Tier, and the Enterprise Information System (EIS) Tier. The Client Tier holds general-purpose clients (e.g., HTML or WML browsers) as well as regular GUI applications. The Web Tier is made up of JSP and servlets. The Business Tier is the domain of the Enterprise JavaBeans (EJB). The EIS Tier contains databases and legacy systems. J2EE also includes APIs that can be used to access databases (JDBC), process XML (JAXP, etc.), use a naming and directory service (JNDI), and access a messaging service (JMS), among other things.

Figure One shows a scenario involving JSP and all the tiers. A user fills out a form in a Web browser (Client Tier) and submits it. The request is received by a servlet (Web Tier), that validates the input (date and number formats, for instance). It then locates an EJB component (Business Tier) responsible for processing this type of request. The EJB accesses a database (EIS Tier) that returns a result (e.g. a list of items retrieved from the database). When the servlet gets the result, it passes it on to the JSP page, which merges the dynamic data with static markup (a navigation bar, a common header and footer, etc.) and sends back the complete response to the browser. The communication between the servlet and the JSP is based a mechanism defined by the servlet specification: the `RequestDispatcher` and request attributes. You can read more about the `RequestDispatcher` in the servlet specification (http://java.sun.com/products/servlet/).

So, do you need a complete J2EE environment to use JSP? Not at all. For a simple application, such as a searchable employee directory, a JSP that accesses a database using a custom action fits the bill perfectly. More complex applications often use a servlet for the request processing and JSP for the user interface. This gives you the best of both worlds: a programmer can use the full power of Java and APIs for request processing in the servlet, and a Web designer can design the site using JSP to add dynamic content. As long as you use only JSP and servlets, all you need is a Web container and the Java 2 Standard Edition (J2SE) environment.

Deploying your JSP

To deploy a JSP on your system you need a Web server that supports the JSP specification. Apache Tomcat 4 is the reference implementation for the JSP 1.2 and the Servlet 2.3 specifications and works great as a JSP development server. For more information on setting up Tomcat, see "Hangin' With Tomcat" in the October 2001 issue (http://www.linux-mag.com/2001-10/tomcat_01.html).

Once Tomcat is installed, adding a JSP is as simple as placing it in a directory underneath *$CATALINA_HOME/Webapps* (remember, Tomcat expects to find applications in the *Webapps* directory and treats it as though it were equivalent to the document root). For example, you could copy our JSP file *foo.jsp* to *$CATALINA_HOME/Webapps/examples/jsp/mytest/foo.jsp* and then access it by browsing the URL http://localhost:8080/examples/jsp/mytest/foo.jsp. Note that you do not necessarily need to do anything with the *web.xml* file, unless you need the special functionality it provides (see last month's column on servlets at http://www.linux-mag.com/2002-04/java_01.html for more on the *web.xml* file).

JSP elements are used to control the dynamic portions of a Web page.

Tomcat 4 comes with JSP examples that you can play with, as well as JavaDocs for both the JSP and Servlet APIs. The JSP examples are located in a Web application (directory) named *examples*, organized according to the standard layout defined by the servlet specification. This layout was described in detail in last month's column.

By now, you're hopefully beginning to see how the different pieces of the Java puzzle fit together and you can start to experiment on your own. A useful next step would be to further learn more about taglibs, which are covered in the extensive Tomcat documentation and on the Jakarta site as well (http://jakarta.apache.org/taglibs/). You should also check out the JSP Specification, Sun's tutorial (http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro.html), and the JSPInsider (http://www.jspinsider.com/index.view) to learn more about JSP.

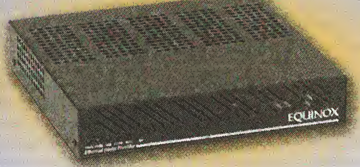
Hans Bergsten is a member of the working groups for the Servlet, JSP and JSTL specifications, and the author of O'Reilly's JavaServer Pages *book. You can reach him at hans@gefionsoftware.com.*

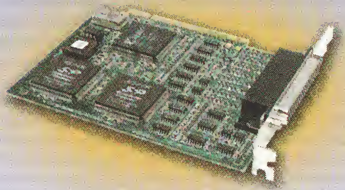
Serial Connectivity Solutions

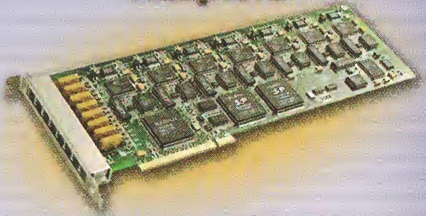
ESP Family of 10/100 Serial Hubs

Multiport Serial Adapters
2-128 ports per server slot

PCI Multi-modem Adapters







"Serial ports over IP"
Place 8 or 16 serial ports at the point of need.

These bus-based adapters provide leading-edge, high performance serial connectivity for a wide range of applications.

Dial access at its best!
Adapters available in 4 and 8 port models.

Reliable and Cost-effective Serial Connectivity Solutions

Mention this ad and qualify for a FREE 30-day evaluation.
Call today, 800-275-3500 ext: 603 or email: sales@equinox.com
For more information visit www.equinox.com

EQUINOX®
an Avocent Company

Are Your Pipes Clogged?

By Randal Schwartz

Recently, I attended a presentation at the Portland Linux Unix Group (http://www.pdxlinux.org/) by Michael Rasmussen. At one point in his talk, Michael mentioned that he needed to analyze the traffic on his company's Web server and was surprised that many of the commercial and freely available Web traffic tools do not provide satisfactory reports on the amount of traffic, either bytes per second or hits per second, during "bursts" or "spikes" in the load. But, being reasonably fluent in Perl, Michael wrote a quick script to crawl through the text Web log, and got the data he needed.

That got me thinking about how much (or little) traffic I was shooting out of the Web server for www.stonehenge.com. Since I'm already storing the data into a MySQL database (via DBI and `mod_perl`), I thought it shouldn't be tough to analyze not only the bytes and hits per second (average and burst) as he had done, but also the CPU usage per second, which I'm also recording for each hit in my log.

To make the result more realistic, I decided to spread out the total bytes and CPU usage over the duration of the hit, which I'm also recording as wall-clock seconds. For example, if 100K has been downloaded over 10 seconds, each second in the span should get 10K of that allocation on average. And because this is a quick-and-dirty program to get some typical stats, I decided to handle this by simply having an array with a separate slot per second, limiting the reasonable size of analysis to just a few days. But that was enough to cover a useful time period, and actually gave me some information about usage that I didn't have from my other reports.

So, let's look at how I did all that in *Listing One*.

Lines 1 through 3 start nearly every program I write, enabling warnings and compiler restrictions, and disabling the normal buffering of standard output.

Line 5 brings in the standard DBI module. For this program, I also had to have the `DBD::mysql` module installed to access the database. Line 6 pulls in the `Statistics::Descriptive` module, so I don't have to remember how to do the mean and max values, and because I was also considering the standard deviation (but ended up not using it). All these modules are found in the CPAN.

Lines 8 through 10 provide the configuration constants. In particular, the `$DSN` value defines the DBI connection information, while `$DB_AUTH` gives the username and password for the particular database. Obviously, these values have been slightly smudged for security reasons. And `$DAYS` gives the number of days (an integer) over which this report will process. We're making and processing three arrays of this value times 86400 (seconds in a day), so don't get too eager to crank it up to a large value.

Lines 12 and 13 set up the database connection, including enabling automatic "die on error" handling, so I don't have to check the valid return status and every database call. I routinely enable MySQL's big table mode, because I've been burned when I didn't do this in the past.

Line 14 is for debugging. When I'm not sure what I'm passing to DBI, or what values are getting returned, I can uncomment this line to get a tracing of operations.

Lines 16 to 21 prepare the SQL query to fetch the data of interest. The return value is a "statement handle," that

LISTING ONE - PART 1

```
1   #!/usr/bin/perl -w
2   use strict;
3   $|++;
4
5   use DBI ();
6   use Statistics::Descriptive;
7
8   my $DSN = 'dbi:mysql:stonehenge_httpd';
9   my $DB_AUTH = 'do_not:try_this_at_home';
10  my $DAYS = 2;
11
12  my $dbh = DBI->connect($DSN, (split ':', $DB_AUTH),
                            { RaiseError => 1 });
13  $dbh->do("SET OPTION SQL_BIG_TABLES = 1");
14  ## $dbh->trace(2);
15
16  my $sth = $dbh->prepare(qq{
17    SELECT unix_timestamp(when), wall, bytes, cpuuser
        + cpusys + cpucuser + cpucsys
18    FROM requests
19    WHERE when >= DATE_SUB(NOW(), INTERVAL $DAYS DAY)
20    ORDER BY when
21  });
22
23  my $offset = time - 86400 * $DAYS;
24
25  $sth->execute();
26  $sth->bind_columns(\my($when, $wall, $bytes, $cpu));
27
28  my @hits, @bytes, @cpu;
29
30  while ($sth->fetch) {
```

allows me to carry out the calculation and obtain the results. The parameter is a double-quoted string, using the `qq{...}` notation so I can easily balance the brackets nicely with my text editor. The one value that is interpolated is the number of days to fetch backwards in time. MySQL's date functions are a bit odd to wrangle, and took me a few tries to get right.

My Web server log is stored in the table `results`. In that table, the `when` column is a date/time stamp of the transaction, `wall` is the wall-clock seconds (as an integer) from start to end, `bytes` is the number of bytes returned by the server, and the four `cpu` values represent the user, system, child user, and child system CPU seconds (as floating point).

The query returns the Unix timestamp value (seconds since the beginning of 1970 in London), the wall clock seconds, the number of bytes, and the sum of the CPU usage in seconds, for all entries that have been posted in the past `$DAYS` number of days. The result is ordered for no apparent reason, except that during testing I was adding things like `LIMIT 10` and wanted a consistently understandable output.

Line 23 computes the Unix timestamp corresponding to the beginning of the period in question. Note that if the time clocks of the DBI server and the server this process runs on are out of sync, this is probably not going to work. Luckily, I was actually doing this on the very same machine, so there's no chance that the values were out of sync.

Line 25 starts the database query humming, and may even fetch all the results into a temporary area.

Line 26 declares the result to have four output values for the columns, which will be stored directly into the four named scalars as I fetch each row of the result. This keeps the interface simple, as well as very fast.

Line 28 declares the three accumulating arrays, for number of hits, total bytes transferred, and total CPU seconds used. These arrays will grow to be 86400 times `$DAYS` elements each. Since `@bytes` and `@hits` are always integers, I considered playing with a `vec()`-based data structure, but decided to optimize along those lines only if the straight solution didn't fit in memory. Never optimize prematurely, especially when you're facing deadlines.

Lines 30 to 40 process the database response data. As each row is returned (indicated by a true value back from `fetch`), the values are placed directly into the four scalar variables named in line 26. The next step is to apportion the hits, bytes, and CPU seconds across the wall-clock interval. First, line 31 computes the initial array index. Line 32 scales down

Never optimize prematurely, especially when you're facing deadlines.

the one hit we're handling by 1 more than the wall count. Since we're counting integer seconds, we've actually got a rounding error here, but a kludge of simply flattening out the request over a slightly larger rounded-up time period was satisfactory to me. And the +1 prevented a potential divide-by-zero error nicely.

Lines 33 and 34 adjust the bytes and CPU seconds to their per-second values, making it easier for lines 35 to 38 to accumulate that value across the spread of the range of seconds for this hit. I actually don't like the repetition of the code: I'm saying the same thing with different variables far too often here, and could probably have factored that out with a bit more effort.

Once the database scan is complete, we disconnect in line 42 to let the server know we're done there.

Lines 44 through 46 fix the `undef`s that appear in the

See Perl of Wisdom, *pg. 63*

LISTING ONE - PART 2

```
31     $when -= $offset;
32     my $hits = 1 / ($wall + 1);
33     $bytes *= $hits;
34     $cpu *= $hits;
35     for ($when..($when + $wall)) {
36       $hits[$_] += $hits;
37       $bytes[$_] += $bytes;
38       $cpu[$_] += $cpu;
39     }
40   }
41
42   $dbh->disconnect;
43
44   defined $_ or $_ = 0 for @bytes;
45   defined $_ or $_ = 0 for @hits;
46   defined $_ or $_ = 0 for @cpu;
47
48   if (0) {                              # dump raw data
49     for (0..$#bytes) {
50       printf "%30s %10.1f %4.1f %4.1f\n",
51         scalar(localtime($offset+$_)), $bytes[$_],
           $hits[$_], $cpu[$_];
52     }
53   }
54
55   for my $pair ([bytes => \@bytes], [hits => \@hits],
                   [cpu => \@cpu]) {
56     my $stat = Statistics::Descriptive::Full->new;
57     $stat->add_data(@{$pair->[1]});
58     for my $thing (qw(mean max)) {
59       printf "%4s %5s/sec %10.3f\n", $thing, $pair->[0],
           $stat->$thing();
60     }
61   }
```

Top Tech Support Questions

By Quentin Cregan

1. How Can I Quickly Tell What File Systems My Current Kernel Can Handle?

The kernel provides an up-to-date list of all the file system types that it's able to mount in the */proc* file system. To take a look at this list, simply run the command `cat /proc/filesystems`. You'll get output that will look something like *Figure One*.

In this output, the entry `vfat` means you can mount FAT/VFAT (Microsoft Windows) partitions. The entries ending with `smbfs` and `nfs` mean you can interact across the network with file servers that use SMBFS (Microsoft's Server Message Block File System, accessed via Samba) or NFS (Sun's Network File System). The `iso9660` indicates that you can mount standard CD-ROM file systems, and `ext3` and `ext2` indicate that you can mount those kinds of Linux file systems.

In the first column, `nodev` indicates that the file system is not associated with a physical device, like the `proc` file system itself, which has information about state of the running kernel.

2. What Is File System Journaling, and How Do I Install It?

File system journaling is a method of ensuring that all the data on a file system is correct and uncorrupted. Any time a change to a file occurs (such as a change in the contents and/or a change in information about the file itself, such as its permissions), that change is not actually written directly to the file system. Instead it is sent to a special location on the disk called the "journal". Every so often, these changes are taken from the journal ("trimming the journal") and written to the disk. A special flag is used to signal that a change is occurring (i.e., data is being written), and that flag is cleared when the entire change has been written (i.e., the change has been "committed").

FIGURE ONE: Contents of */proc/filesystems*

```
nodev   proc
        ext3
        ext2
        vfat
        iso9660
nodev   nfs
nodev   smbfs
```

The primary benefit of journaling is that in the event of a system failure (such as a power outage), only those files whose changes were in the process of being committed need to be checked to make sure they are in a coherent state. The last few changes that weren't fully written to the disk (i.e., weren't committed) need to be "replayed" to restore the file system to a consistent state.

Two popular journaling file systems are ext3 and ReiserFS.

This means that rebooting after a system failure doesn't require a full file system check (via the *fsck* program), which can be excrutiatingly slow on a large disk. For more information on journaling file systems, consult the August 2000 issue online at http://www.linux-mag.com/2000-08/journaling_01. html.

Two journaling file systems that have become popular recently are ext3 and ReiserFS. Ext3 is compatible with ext2 file systems, which are the default for many Linux distributions today. Ext3 is basically ext2 with journaling. It's considered more "well rounded" than the ReiserFS, which can be faster at times.

To convert your existing ext2 file systems to ext3, your kernel must have ext3 support built into it. It comes standard in kernels 2.4.16 and later, and Andrew Morton has made patches available for 2.4.9 and later at http://www.zip.com.au/~akpm/linux/ext3/. There are also detailed installation instructions, including links to other required downloads (`e2fsprogs` and `util-linux`). If you're using a 2.2-series kernel, you'll want to upgrade, because the 2.2 patch has not been maintained since June 2001.

Once the kernel is ready for ext3 (and the system has been rebooted if you rebuilt your kernel), run the `mount` command to see the mounted file systems. Your output will look something like this:

```
/dev/sda3 on / type ext2
  (rw,errors=remount-ro,errors=remount-ro)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts
  (rw,gid=5,mode=620)
/dev/sda1 on /boot type ext2 (rw)
/dev/sda2 on /usr type ext2 (rw)
/dev/sda5 on /var type ext2 (rw)
/dev/sdc1 on /dump type ext2 (rw)
```

Let's pick the */dump* ext2 partition on */dev/sdc1* and convert it. First, run (as root) the command `tune2fs -j /dev/sdc1` to create the journal file. This should only take a second.

When this is done, edit the entry for that file system in */etc/fstab* and change `ext2` to `ext3`. That will make it look something like:

```
/dev/sdc1 /dump ext3 rw 0 2
```

You may be able to simply unmount and then remount the partition. Otherwise, you'll need to reboot. Once restarted, you have a mounted ext3 file system and are free of those long file system checks!

You should note that about every 20 mounts of the file system, or once every 180 days on average, an automatic consistency check will be done, which entails running the *fsck* program. If you're an average user, you should let the system check it automatically and just live with the delay when it occurs.

If you're a production sysadmin and want to have better control over when the consistency check occurs, you can disable the automatic check by running the following command (replacing the device file name with the appropriate one):

```
# tune2fs -i 0 -c 0 /dev/sdc1
```

However, you will then need to schedule downtime to run *fsck* manually.

3. How Can I Get More Information About Files and Executables?

There is a standard utility that comes with most distributions called `stat`. The `stat` command allows you to quickly view standard metadata about files on your system. This is useful in debugging, or getting file-aging information. It is also useful for finding out when a file was last edited or accessed.

`stat` outputs all of the information it can display about a single file, even if you don't have permissions to read the file; you must have at least list permission on the directory in which the file lives (denoted by `x` in the output of `ls -l`).

To use `stat`, simply use the file name you're interested in as an argument. If you run `stat picture.jpg`, you might see output like that in *Figure Two*.

FIGURE TWO: Output from stat

```
$ stat picture.jpg
  File: "picture.jpg"
  Size: 148176          Blocks: 304      IO Block: 4096    Regular File
Device: 821h/2081d      Inode: 231         Links: 1
Access: (0600/-rw-------)  Uid: ( 1000/       qc)    Gid: ( 1000/       qc)
Access: Mon Feb 12 22:55:21 2001
Modify: Mon Feb 12 22:19:57 2001
Change: Mon Feb 12 22:19:57 2001
```

FIGURE THREE: Output from file

```
$ file *
Mail:           directory
column:         English text
export:         directory
foo.pl:         perl script text
inv2:           ASCII text
legend1.gif:    GIF image data, version 87a, 573 x 391
mail:           symbolic link to Mail/
master.zip:     Zip archive data, at least v2.0 to extract
postponed:      empty
```

Another useful tool is `file`. The `file` command identifies the type of file whose name is provided as an argument. It can identify well-known types of files and provide information about the data stored in them.

For example, running the command `file picture.jpg` might return output like:

```
$ file picture.jpg
picture.jpg: JPEG image data,
  JFIF standard 1.02, resolution (DPI),
  72 x 72
```

The `file` command can also return information about an executable:

```
$ file netwatch
netwatch: ELF 32-bit LSB executable,
  Intel 80386, version 1 (SYSV),
  dynamically linked (uses shared libs),
  not stripped
```

You can also run `file *` to get information for all of the files in your local directory. That output might look like that in *Figure Three*.

Quentin Cregan is a tech and security consultant and sometime student hailing from Australia. He can be reached at qc@sensed.com.

Zonker's Product Picks

The Best and the Coolest New Linux Products. *By Joe "Zonker" Brockmeier*

SNAPGEAR SOHO+ VPN ROUTER: SECURITY IS A SNAP

Need to set up a Virtual Private Network (VPN) between two remote offices, but don't want to struggle with configuring the software? SnapGear has just the thing for you.

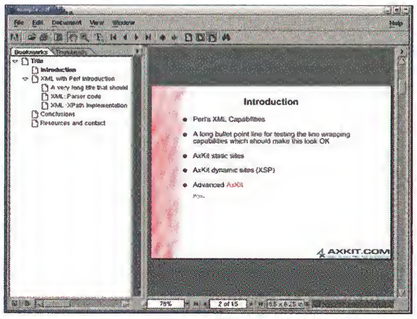
SOHO+, one of SnapGear's VPN appliances, is powered by uClinux. It handles unlimited users, DHCP, Network Address Translation, Port Forwarding, Traffic Shaping, VPN over IPSec, and PPTP (Point-to-Point Tunneling Protocol).The SOHO+ can handle VPN traffic at 1-2Mbps, an ideal solution for small businesses. The suggested price for the SOHO+ is $399. For more on SnapGear VPN hardware, go to http://www.snapgear.com/.

AXPOINT: MAKE YOUR POINT WITH POWER

Want to make a snazzy presentation that anybody can view, whether they're using Linux, Microsoft, Macintosh, Solaris, IRIX, or just about any other platform? AxPoint may be just what the doctor ordered.

AxPoint is a Perl module that converts an XML file into a presentation in Portable Document Format (PDF). AxPoint presentations can include background images and transition effects like dissolves and wipes. Text on slides can be displayed in bold, italics, and color. Slides can also show bullet points, source code (in a fixed font), or images. Axpoint supports the GIF, JPEG, PNG, and TIFF image formats.

AxPoint was written by Matt Sergeant and is available under the Apache license. For more on AxPoint, or to grab your own copy, visit http://axpoint.axkit.org/.



Introduction

- Perl's XML Capabilities
- A long bullet point line for testing the line wrapping capabilities which should make this look OK
- AxKit static sites
- AxKit dynamic sites (XSP)
- Advanced AxKit

GEN2 P4 LINUX LAPTOP: PORTABLE POWER

Pentium 4s are finally appearing in laptops, and you don't have to wait to get your hands on a Linux model.

In addition to a P4, the QliTech Gen2 features a 14.1" XGA display with the ATI 16MB Radeon video chipset. The Gen2 can be loaded with your favorite Linux distribution and includes a one-year parts and labor warranty.

The Gen2 starts at $1659 with a 1.5GHz P4 (upgradable to 2.0GHz), 256MB of RAM, 20GB hard drive, 8x DVD drive, and much more. The machine weighs in at 6.5 lbs and is less than two inches thick. For more on the Gen2 or other Linux machines, peruse QliTech's virtual storefront at: http://www.qlilinux.com/.

ZEND STUDIO 2.0: BROUGHT TO YOU BY THE LETTER "Z"

Put all the resources you need to develop PHP apps at your fingertips with the Zend Studio package, an all-in-one environment for working with PHP.

The Zend Studio consists of two parts. The first is the Client package, which includes the Zend Development Environment (ZDE) and Information Center. The ZDE features a code completion engine and a PHP debugger. Then there's the Server package which contains the Zend Debug Server, the Zend Optimizer, the Zend Server Center, PHP 4.1.0, and Apache 1.3.22.

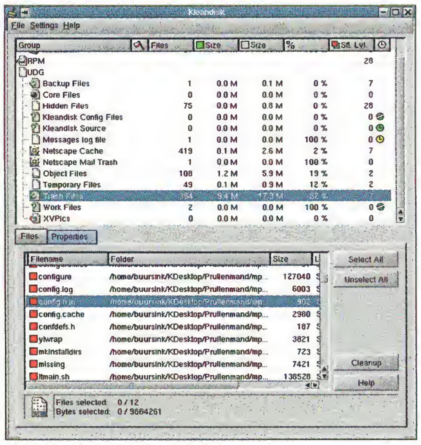
Licenses for Zend Studio start at $195 per seat, with discounts for volume purchases or academic licenses. The price includes one year of updates and support. For more on Zend Studio visit http://www.zend.com/.

KLEANDISK: KEEP IT CLEAN

Though disk space isn't the problem it used to be, it's still important to conserve on laptops and other machines that might not have it to spare. Kleandisk can take some of the tedium out of managing disk space.

Kleandisk can search for specific file types, clean on a preset schedule, search for duplicate files, and rotate log files. It can also help manage RPMs and archive files before deleting them from your disk. Kleandisk can even be used as a backup and archive tool.

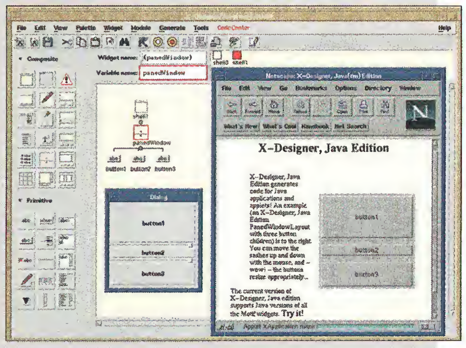
Kleandisk requires KDE 2.0 or later and is available under the terms of the GNU General Public License. Kleandisk is available for most of the popular Linux distributions. To get your copy visit the Kleandisk homepage at: http://home.wxs.nl/~arjan.buursink/.



```
Group                     Files   Size    Size     %      Sft. Lvl.
RPM                                                        28
UDG
  Backup Files               1    0.0 M   0.1 M    0 %      7
  Core Files                 0    0.0 M   0.0 M    0 %      0
  Hidden Files              75    0.0 M   0.8 M    0 %     28
  Kleandisk Config Files     0    0.0 M   0.0 M    0 %      0
  Kleandisk Source           0    0.0 M   0.0 M    0 %      0
  Messages log file          1    0.0 M   0.0 M  100 %      0
  Netscape Cache           419    0.1 M   2.6 M    2 %      7
  Netscape Mail Trash        1    0.0 M   0.0 M  100 %      0
  Object Files             108    1.2 M   5.9 M   19 %      2
  Temporary Files           49    0.1 M   0.9 M   12 %      2
  Trash Files              384    9.4 M  17.9 M   52 %      7
  Work Files                 2    0.0 M   0.0 M  100 %      0
  XVPics                     0    0.0 M   0.0 M    0 %      0

Filename      Folder                                     Size
configure     /home/buursink/KDesktop/Prullenmand/mp...  127040
config.log    /home/buursink/KDesktop/Prullenmand/mp...    6003
config.h.in   /home/buursink/KDesktop/Prullenmand/mp...     902
config.cache  /home/buursink/KDesktop/Prullenmand/mp...    2980
confdefs.h    /home/buursink/KDesktop/Prullenmand/mp...     187
ylwrap        /home/buursink/KDesktop/Prullenmand/mp...    3821
mkinstalldirs /home/buursink/KDesktop/Prullenmand/mp...     723
missing       /home/buursink/KDesktop/Prullenmand/mp...    7421
ltmain.sh     /home/buursink/KDesktop/Prullenmand/mp...  138528

Files selected:  0 / 12
Bytes selected:  0 / 9664261
```

X-DESIGNER 7.0: X MARKS THE SPOT

Looking for a Rapid Application Development tool for X/Motif applications? What about cross-platform X and Windows apps? X-Designer could be just what you're looking for.


Netscape: X-Designer, Java(tm) Edition
File Edit View Go Bookmarks Options Directory Window
What's New What's Cool Handbook Net Search
button1
button2
button3

X-Designer, Java Edition

X-Designer, Java Edition generates code for Java applications and applets! An example (an X-Designer, Java Edition PanedWindowLayout with three button children) is to the right. You can move the sashes up and down with the mouse, and – wow! – the buttons resize appropriately...

The current version of X-Designer, Java edition supports Java versions of all the Motif widgets. Try it!

X-Designer can generate code in C, Java, C++ for X/Motif, C++ for Microsoft Foundation Classes (MFC), and Ada 95. It runs on Linux, Solaris, AIX, and a number of other versions of Unix. X-Designer lets you build test scripts to put your applications through the paces before deployment. It can also generate Makefiles.

X-Designer is a product of Imperial Software Technology; single-user licenses start at $4,500. For information, visit http://www.ist-inc.com/.

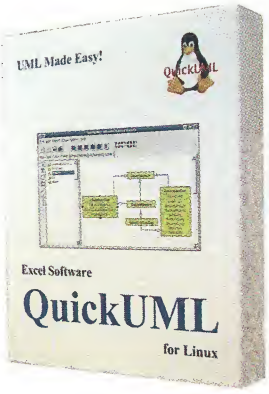
MORPHON XML EDITOR: EDITING XTACY

The eXtensible Markup Language is just about the greatest invention since sliced bread. It's portable and can be used to produce documents in just about any format. Unfortunately, most people don't enjoy writing documents directly in XML. The Morphon XML Editor takes all the pain out of creating XML documents by bringing a near-WYSIWYG interface to just about any XML Document Type Definition (DTD) or XML Schema.

Morphon runs on any platform with a compliant Java Runtime Environment, so you can use it just about anywhere. You can create documents based on a variety of XML DTDs or Schemas, including DocBook. Morphon also supports extensions through a plug-in API, so if it doesn't have a feature you'd like to see, you can add it yourself.

Prices for the Morphon XML Editor starts at $150 for a single-user license, with student and volume discounts available. For more information visit the Morphon Technologies Web page at http://www.morphon.com.

QUICKUML: MODELING MADE EASY



Excel Software's QuickUML is a cross-platform tool that supports the Unified Model Language, which has become the industry-standard notation for object-oriented analysis and design. Use QuickUML to create object-oriented designs in multiple languages, view code in a separate window while creating UML diagrams, and link use cases to code files. QuickUML saves its projects in XML and can share information with other Excel Software programs like QuickCRC or WinA&D.

QuickUML requires 64MB of RAM and 100MB of disk space. The single license price for QuickUML is $495, with five-user and unlimited site licenses available. Find out more about QuickUML at http://www.excelsoftware.com/.

Got a product pick of your own for Zonker? Drop him a line at zonker@linux-mag.com

IPv6, *from pg. 36*

specified with any address by appending "`%`" and the interface name (for example, `ping6 ff02::1%eth0`).

The Domain Name System (DNS) maps host names to IP addresses. A DNS server normally returns IPv4 addresses when given a host name (this is a query for an `A` record). To ask the server to return IPv6 addresses, you must make a query for either an `AAAA` record (which is older and more established) or an `A6` record (which is newer and hasn't yet gained widespread acceptance). You can test the ability of your DNS servers to resolve IPv6 addresses by running the command:

```
# host -t AAAA www.6bone.net
www.6bone.net. is an alias for 6bone.net.
6bone.net. has AAAA address 3ffe:b00:c18:1::10
```

Figure Four (pg. 34) shows the output of some IPv6 utilities. This includes *ping6* to test IPv6 packet transmission to the 6bone, *traceroute6* to see how the connection is made, and *tracepath6*, which gives information similar to *traceroute6*.

If you still have problems, you can also use the packet capture program *tcpdump* on the outgoing `eth0` interface (or better, on all interfaces you have access to) to see what's going on. Releases of *tcpdump* since version 3.6 are IPv6-enabled. *Figure Five* (pg. 35) some of the most useful features for debugging IPv6 problems.

IPv6 CLIENTS

The current versions of many Web browsers are IPv6-enabled (*Mozilla*, *w3m*, *Netscape 6*, *mMosaic*, *lynx*, and *Konqueror*).

A good starting point for browsing is http://www.kame.net/. If the turtle on this page is animated, the connection is via IPv6; if it's static, the connection is via IPv4 (you have to use a GUI browser for this test). Another URL for testing is http://ipv6.aerasec.de/. Near the bottom of the screen it will read, "Your connection is via: IPv6."

Other clients already IPv6-enabled include Telnet, Secure Shell (SSH), and FTP. Most applications will recognize an IPv6 address as an argument, or if given a host name, will try to resolve it first as an IPv6 address and then fall back to IPv4 if that doesn't work. But be careful if you are using a proxy relay, because most cannot resolve IPv6 requests. See the man pages for more information.

IPv6 SERVERS

We're finally ready to look at Apache. Many server applications are already IPv6-enabled, but IPv6 is seldom switched on by default. Before you launch any IPv6-enabled services it's always a good idea to add a static IPv6 address on the related interface. Because an IPv6-enabled interface can handle multiple addresses, you can do this easily.

For example, to add the address `2002:d151:90f::5` on `eth0`:

```
# /sbin/ip -6 addr add \
  2002:d151:90f::5/64 dev eth0
```

You can also remove existing addresses using `del` instead of `add`.

Now let's enable IPv6 on an Apache server. You need to be running at least version 2.0.32. We'll assume you're using virtual hosts, because the configuration files are easier to change. You can set up two kinds of virtual hosts (see *Figure Six* pg. 35). Note that because the port address is separated from the IP address by a colon, the IPv6 address must be placed within square brackets to separate it from the port number.

After restarting the server, you should be able to run the test in *Figure Seven* (pg. 35) to confirm that Apache is working. You should now be able to test connectivity from your your IPv6-enabled browser.

LOOKING AHEAD

We've only scratched the surface of the brave new world of IPv6. As you move ahead, you'll want to look at setting up a Domain Name System using BIND.

Also, if you want to set up IPv6 for all your clients on a link, you need to control only one box (the IPv6 router). After installing the Router Advertisement Daemon on it, this daemon announces information about default gateway and address prefixes to use. You can think of this as DHCP on steroids, although DHCPv6 is still a work in progress.

Like IPv6 itself, IPv6 in Linux is still incomplete. The good news is that in current distributions, basics like routing, tunneling, autoconfiguration, and client and server applications are working well. The not-so-good news is that special features and enhancements are not yet in the mainstream.

Dr. Peter Bieringer is the author of the Linux IPv6 HOWTO. *He works for a network and security consulting company near Munich, Germany, and can be reached at pb@bieringer.de.*

RESOURCES

6bone information
http://www.6bone.net/

Switch (a big list of IPv6 references)
http://www.switch.ch/lan/ipv6/references.html

IPv6 HOWTO (Linux Documentation Project)
http://linuxdoc.org/HOWTO/Linux+IPv6-HOWTO/

Author's IPv6 Site (includes HOWTO and other information)
http://www.bieringer.de/linux/IPv6/

Power Tools, *from pg. 39*

Alternatively, if you don't want to create your own prompt, you can use a prepackaged prompt by adding the following to your *.zshrc* file:

```
autoload -U promptinit
promptinit
prompt bart
```

You can replace "bart" with the name of any prompt that you like. To see a list of choices, type `prompt -l`, or `prompt [TAB]` to use tab-completion to get the list of prompts at your disposal. Even if you decide to make your own custom prompt, the prepackaged prompts are a good resource for learning how prompts in *zsh* work.

To examine and modify the `$PS1` variable (or any other variable, for that matter), you can use `vared`:

```
$ prompt suse
beppu@thestruggle:~/ > vared PS1
%n@%m:%~/ >
```

At this point, `vared` will let you use your arrow keys to move around and edit this variable. When you're done, hit `[ENTER]` and your changes will take effect.

Everything Else

So far we've covered areas where *bash* and *zsh* are different, highlighting some of the unique features of *zsh*. However, *bash* and *zsh* have a lot in common. In fact, much of your *bash* configuration can be copied straight into your *zsh* configuration with few or no changes.

Aliases should be completely compatible, and functions should be fine if you limited yourself to basic Bourne Shell semantics. If you've used any advanced features of *bash* though, you may have some work to do to get them to function in *zsh*. And finally, environment variables can be defined and exported exactly the same way. This makes the transition from *bash* to *zsh* relatively easy.

If you've liked what you've seen so far, there's just one thing left to do. Run `chsh` and set *zsh* as your login shell (making sure to define the full path to *zsh*). The next time you log in, you should be greeted by a *zsh* prompt.

With *zsh* you should have all the functionality that you had in *bash* and more. However, we've only scratched the surface as far as what *zsh* is capable of, so when you get a chance, Read The Fine Manual to uncover more of what *zsh* has to offer.

John Beppu beppu@cpan.org encourages you to use your computer skillfully and creatively.

Guru Guidance, *from pg. 41*

hierarchy has both a name and a number associated with it and can be referred to by either. The numbers for each item are shown in parentheses in *Figure Two*. For example, `iso.org.dod.internet.mgmt.mib-2.system.sysLocation` can also be referred to as 1.3. 6.1.2.1.1.6. Similarly, the `laLoad` data can be accessed as `iso.org. dod.internet.private.enterprises.ucdavis.laTable.laEntry.laLoad` or as 1.3.6.1.4.1.2021.10.1.3. Each of these name types is known generically as an OID (object ID). Sometimes, only the name of the final data node (`sysLocation` or `laLoad`) is needed to refer to a data node, but sometimes the full "path" of the OID must be specified.

The SNMP Community

Access to SNMP data is controlled by passwords that are called "community" names or strings. There are generally separate community names for the agent's read-only and read-write modes, as well as an additional name used for traps. Each SNMP agent knows its name (or password) for each mode, and will not answer queries that specify anything else. Community names can be up to 32 characters long and should be chosen using the same security considerations as root passwords (not easily guessable or crackable). We'll discuss security issues in more detail next month.

Unfortunately, many devices are shipped with SNMP enabled, and have the read-only community name set to "public" and sometimes the read-write community name set to "private". It's important that you change these values (or at least disable SNMP for the device) *before* the device is added to the network. Otherwise, it is vulnerable to attack by hackers and can put other parts of your network at risk.

The way to change these values varies by device. For hosts, it is changed in the configuration file associated with the SNMP agent (we'll examine the one for Linux systems when we actually configure an agent next month.). For other devices, such as routers, consult the documentation provided by the manufacturer.

In contrast to the complex data definitions, the set of SNMP operations that monitor and manage devices is quite limited, consisting of `get` (requests a value from device), `set` (specifies the value of a modifiable device parameter) and `trap` (sends a trap message to a specified manager).

Next month we'll install an SNMP package that allows a Linux host to function as an SNMP agent that also includes commands to print the definition of nodes in a MIB as well as perform the above SNMP operations.

Æleen Frisch is the author of Essential System Administration. She can be reached at aefrisch@lorentzian.com

Extreme Linux, *from pg. 43*

In *Listing One* (pg. 43), the matrices **A** and **B** are read from files (lines 23-45), and then `MPI_Scatter()` is used to send a column of **A** and a row of **B** to each process (lines 47-53). Because we know that **C** stores matrices (arrays) in memory a row at a time, we'll send a column of **A** by building its transpose (rows and columns are switched) as we read in the elements (line 30), and send a row of the transpose. This will correspond to a column of **A**.

In lines 47-48 and lines 51-52, `MPI_Scatter()` is passed the address of the data buffer to send (`atrans` and `b`); the number elements to send to each process (`M` and `P`); the data type being sent (`MPI_FLOAT`); the address of the receive buffer (`avec` and `bvec`); the number of elements to receive (`M` and `P`); the data type to receive (`MPI_FLOAT`); the rank of the sending process (`0`); and the communicator (`MPI_COMM_WORLD`).

Next each process calculates its own product for each element of **C** in lines 55-57. These products must be summed across all processes in order to arrive at the result. `MPI_Allreduce()` is used to accomplish this in lines 59-60. Like `MPI_Reduce()`, `MPI_Allreduce()` performs an operation (again, these can be pre-defined or user-defined) on the data from each process. It's passed the address of the send buffer (`c`), the address of the receive buffer (`d`), the number of elements to send (`M*P`), the data type (`MPI_FLOAT`), the operation (`MPI_SUM`), and the communicator (`MPI_COMM_WORLD`). Now every process has the answer stored in the array `d`.

The program is compiled and run as shown in *Figure Two*. Sure enough, it works!

The Art of (Parallel) Programming

Obviously, you wouldn't use a parallel computer to solve such a tiny matrix multiplication problem, but this small problem is perfect for testing an algorithm which may now be scaled up and applied to very large matrices. Careful thought and considerable creativity are required to create scalable parallel algorithms; as has been said before, the process is as much art as it is science.

These last three moth's worth of columns have provided a thorough introduction to parallel programming techniques using the MPI and PVM message passing interfaces. This knowledge will serve you whether you're writing code for a small Beowulf cluster or for the largest commercial parallel computers on Earth.

Forrest Hoffman is a computer modeling and simulation researcher at Oak Ridge National Laboratory. He can be reached at forrest@esd.ornl.gov

Advertisers' Index

The Advertisers' Index lists each company's Web address and advertisement page. To advertise in Linux Magazine, please contact adsales@linux-mag.com for a media kit containing an editorial schedule, rate card, and ad close dates.

ASL Workstations	http://www.aslab.com	11
Apple	http://www.apple.com	C2
Avocent	http://www.avocent.com	13
Consensys	http://www.raidzone.com	C3
Cyclades Corporation	http://www.cyclades.com	7
Equinox	http://www.equinox.com	53
Hewlett-Packard	http://www.hp.com/linux	37
Knox Software	http://www.arkeia.com	21
Microsoft	http://www.microsoft.com/isp	14
Microway	http://www.microway.com	2
PGI	http://www.pgroup.com	47
QLITech Linux Computers	http://www.qlitech.com	18
Red Hat	http://www.redhat.com	C4
SuSE	http://www.SuSE.com	31
Sun Mircosystems	http://www.sun.com	5

Linux Magazine (ISSN 1536-4674) is published monthly by InfoStrada LLC at 234 Escuela Avenue #64 Mountain View, CA 94040. The U.S. subscription rate is $29.95 for 12 issues. In Canada and Mexico, a one-year subscription is $59.95 US. In all other countries, the annual rate is $89.95 US. Non-US subscriptions must be pre-paid in US funds drawn on a US bank. Application to Mail at Periodicals Postage Rates is Pending at Mountain View, CA and additional mailing offices.

POSTMASTER: Send address changes to Linux Magazine, P.O. Box 55731, Boulder, CO 80323-5731.

Article submissions and letters should be e-mailed to editors@linux-mag.com. Linux Magazine reserves the right to edit all submissions and assumes no responsibility for unsolicited material. Subscription requests should be e-mailed to linuxmag@neodata.com or visit our Web site at www.linuxmagazine.com.

Lamp Post, *from pg. 49*

Try it Out

Click on the *Go to your PostNuke site* link to go to the home page for your PostNuke site (the URL would be http://localhost/nuke/index.php). By default, it should look like *Figure Four.*

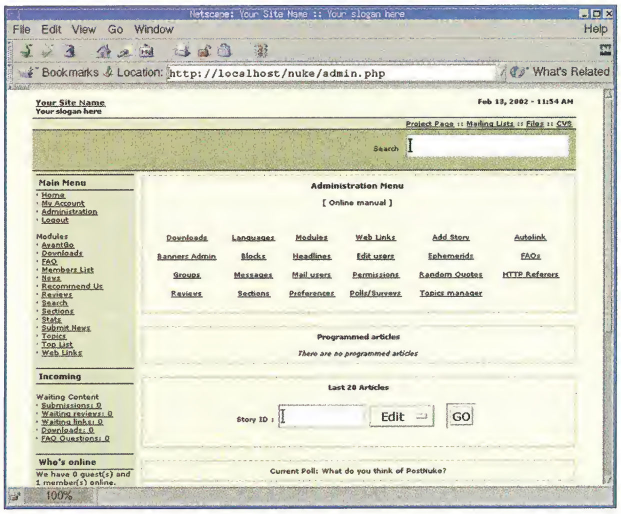
You'll notice that PostNuke is rather generic right out of the box. That's because it was built to be customized. The page title, slogan, poll, and even the color scheme can be adjusted easily. You'll just need to spend some time tinkering with it.

To customize your site, start by entering the administrative username and password and click *Login*. Once you're logged in with administrative rights, you'll see an *Administration* link on the main menu on the upper-left. That link will take you to the Administration Menu, pictured in *Figure Five*.

The Administration Menu is the starting point for customizing all aspects of the site. It includes 23 different configuration settings that control user and group administration, the modules which appear on the home page, news topics, polls, and other site features.

Plugging in Next Month...

With PostNuke installed and running, you can begin adding content and customizing the site's appearance. You'll find



There are no programmed articles

We have 0 guest(s) and 1 member(s) online.

Current Poll: What do you think of PostNuke?

FIGURE FIVE: The PostNuke Administration Menu

that PostNuke is easy to learn and customize, and is also quite extensible. In next month's column we'll look at finding, installing, and using PostNuke plugins to enhance the functionality of your site.

Jeremy Zawodny uses Open Source tools in Yahoo! Finance by day and is is writing a MySQL book for O'Reilly & Associates by night. He can be reached at: Jeremy@Zawodny.com.

Perl of Wisdom, *from pg. 55*

arrays of data for the timespans when there's no activity on my Web server. Leaving these `undef` values in the array will trigger warnings later, even though they are treated as zero. Right result, but noisy, so a quick search-and-destroy takes care of it.

Lines 48 to 53 dump out the raw data in a pretty format. Note that the code is inside an `if (0)` control, so there's no execution of this code in the final program; while I was testing, this code was handy, and I tend to leave my paint scaffolding erected in case there's a chance I'll be doing more painting later. Note the use of the scalar value of `localtime()` to quickly turn a Unix timestamp into a human-readable string.

The real work happens in the final loop, in lines 55 to 61. This loop is executed three times (and yes, it was originally written as a cut-n-paste nightmare where I had the same thing three times with three different variables). Each time through the loop, `$pair` is a reference to a two-element array consisting of a label and a reference to the array that holds the data corresponding to the label. Inside the loop, we create a new statistics object (line 56), push the data into the object (line 57), and then compute the mean and the maximum by calling two different methods against the statistics object. The `mean()` method gets the mean label, and the label of the item being processed from the outer loop. The use of `printf()` here formats the output nicely.

And there you have it. The output for my Web server as I just ran this looks like this:

```
mean   bytes/sec   2014.457
max    bytes/sec   207162.000
mean   hits/sec    0.198
max    hits/sec    25.524
mean   cpu/sec     0.008
max    cpu/sec     1.212
```

At peak, I'm nearly filling a T-1 with the amount of data I'm squirting out, and yet on average, I'm using about 1% of the CPU for Web service. That last value shouldn't be greater than one: there is obviously some discarded data about the specific allocation of CPU to particular seconds during the day, yielding a slightly bogus max CPU seconds per second.

I hope you found it useful to see how to make sure you're filling your pipes with your Web server. Until next time, enjoy!

Randal L. Schwartz is the chief Perl guru at Stonehenge Consulting and can be reached at merlyn@stonehenge.com. Code listings for this column can be found at http://www.stonehenge.com/merlyn/LinuxMag/.

SHUTDOWN

Too Many Servers

By Steven J. Vaughan-Nichols

Remember when the hot machine for running Linux was an Intel 486? Or a Pentium III? For most of us, single-chip Intel PCs are still the computers of choice. It's what we play on, what we work on, and what we develop on.

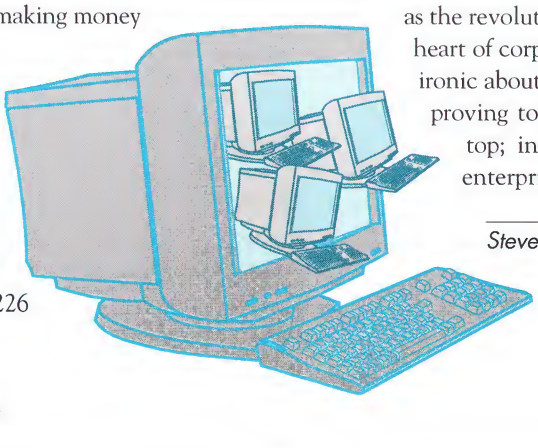
But it seems that more and more the standalone PC or server is not what most Linux users will be running on in the next few years. Oh, there's still a movement toward making Linux a consumer desktop system. Indeed, there's an excellent new Web site devoted to the subject (http://www.desktoplinux.com), and a December 2001 Linux.com poll showed that 25.6% of Linux users already think that Linux *is* a competitive consumer desktop operating system.

But since then, Loki, the one major Linux game company, went out of business. And even before Loki shut down, 17.4% of the people who took that same survey said they think that Linux will never be a big-time operating system for the masses. I'm with them. Sort of. You see, I think Linux will become a brand-name operating system for millions of business users. But they're not going to be running Linux on desktop PCs, they're going to be running applications on a Linux virtual machine (VM) running on a high-powered cluster or on a mainframe.

Who would have thought Linux's fate was to be the center of the enterprise?

You could see signs of this coming change at this January's LinuxWorld. Shirts and ties far outnumbered tie-dyed shirts. The big news of the show wasn't that IBM was pushing Linux on the mainframe: they've been doing that for a while now. And it wasn't their TV ads featuring Linux or their new slogan, "Linux is real business." The big news, according to Bill Zeitler, head of IBM's server group, is that out of the billion dollars IBM poured into Linux, "We've recouped most of it in the first year in sales of software and systems." That's not, "Oh, some day we'll see a return," or "Oh, it's worth it because of good will," that's "IBM is making money from Linux *today*."

This isn't just a retro-mainframe approach. International Data Corp. (IDC) expects Linux clustering, pioneered by TurboLinux, to become the dominant operating system in cluster environments, growing from $226 million in sales for the entire clustering industry in 2000 to $1.4 billion by 2005.



How is it being done? IBM is doing it by running Red Hat, SuSE, and TurboLinux as VMs on their S/390 mainframes and other "big iron" models. VMware is also in the picture with its ESX Server, optimized to run on IBM's Intel-based eServer xSeries systems.

Compaq and Platform Computing are using clustering to follow a similar path on the Alpha. In addition, there are at least four open source VM projects, FreeVSD, Plex86, User-mode Linux (UML), and vserver. Commercial efforts in the same line come from SWsoft and Ensim.

You might be asking, though, "Why bother with virtual machines?" After all, it's not as if your generic Linux box is going to run out of resources for ordinary server programs.

But there are several good reasons to go the VM route. One is security. While the Unix/Linux security model is strong compared to Microsoft's, mainframes are much harder to crack. Even if a cracker does get root access to a single server running as a VM on the mainframe, he's still no closer to getting into other Linux VM servers or the mainframe operating system.

Another advantage VMs have is stability. Half a dozen Linux VM servers can crash, but the other two dozen will still hum along normally. A standalone Intel-based Linux server simply can't compete with that level of stability.

Finally, while Linux is getting better at symmetric multiprocessing (SMP), the IBM mainframe operating systems are much, much better. With the mainframe model, you simply don't have to worry about SMP.

Of course, it's not clear exactly how this will play out. While some are making a move from multiple servers to a few mainframes, Oracle is taking a different approach. In February, CEO Larry Ellison said that Oracle is replacing a trio of Unix servers that run most of its business applications with a cluster of Intel servers running Linux.

What's clear, though, is that Linux's biggest future role isn't as the revolutionary operating system for PCs; it's as the heart of corporate big-iron systems. There's something ironic about that, but it's the price of success. Linux is proving to be too good, too advanced for the desktop; instead its fate is to be the center of the enterprise. Who would have thought it?

Steven J. Vaughan-Nichols is a longtime Unix guru and technology writer. He can be reached at sjvn@vna1.com.

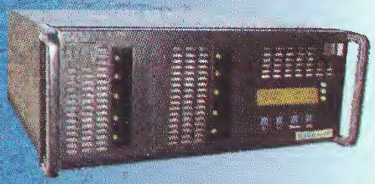
First We Set Price/Performance Records. Now We've Cut RAIDZONE Prices By 40%. Please Buy More Storage!

Network Attached Storage first came under Moore's Law when we invented the RAIDZONE disk array architecture. The largest Ultra ATA disk is now 160 GB, and it doesn't cost much more than the 80 GB drives we were using 6 months ago. We can put twice as much storage into the same chassis, at almost the same cost. That's a 40% price cut in less than a year, with major hardware and software improvements thrown in for free.

Isn't that what happened with PC's?
Rapidly falling prices can cause serious dislocations for old-line vendors. But frankly, we don't care about that and neither do you unless you own their stock. We care about demand elasticity — does the demand for storage increase more rapidly than prices fall? That's what happened with PC's. If there's going to be a "demand tornado" like Intel and Microsoft felt, the same thing has to happen with storage. **We need your help.**

There are many new ways to profitably use inexpensive storage.
Storage prices are lower than anyone ever imagined. You can easily afford to keep online backups, replication, and snapshots of all your data. You can start doing offsite backups and offsite replication over the network. You can install a video security system using online storage. If you work in a hospital, you can store six years of X-rays online instead of six weeks. If you search for oil or smash atoms for a living, you can keep all your data online instead of on tape. Without looking very hard, we can all find new uses for abundant, inexpensive storage that will make our lives happier and more prosperous. At these prices, why not buy a few Terabytes right now while you're thinking about it? Impulse buying, that's what we need!

RAIDZONE RS15



4U 2.0 TB $24,995

RAIDZONE OpenNAS RS15
2 TB File Server, 4U Rack

- 15 hot-swappable, 160 GB Ultra ATA/133 drives (~2. TB free space)
- High performance Dual GigaHz Pentium III CPUs, 512MB ECC RAM

RAIDZONE CS5

Cube 600 GB $8,695

RAIDZONE OpenNAS CS5
600 GB File Server, 14" Cube

- 5 hot-swappable, 160 GB Ultra ATA/133 drives (optional hot spare)
- High performance Dual GigaHz Pentium III CPUs, 512MB ECC RAM

RAIDZONE RS8

2U 900 GB $14,450

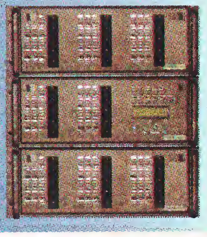
RAIDZONE OpenNAS RS8
900 GB File Server, 2U Rack

- 8 hot-swappable, 160 GB Ultra ATA/133 drives (900 GB free space)
- High performance Dual GigaHz Pentium III CPUs, 512MB ECC RAM

RAIDZONE OpenNAS™ Standard Features

- RAIDZONE "switched node" disk array technology, with RAID 5, hot spare, hot swap, on-the-fly rebuild
- Complete browser-based remote administration interface
- Supports NFS, Windows (SMB/CIFS) and Apple clients
- Runs Linux 2.4 kernel, journaling file system, 4 redundant boot partitions
- Provides snapshots and remote mirroring

RAIDZONE RS15+



8U 4.0 TB $47,490

12U 6.0 TB $67,490

RAIDZONE OpenNAS RS15+
4.0 TB File Server, 8U Rack
6.0 TB File Server, 12U Rack

- 30/45 hot-swappable, 160 GB Ultra ATA/133 drives (~4.0/6.0 TB free)
- High performance Dual GigaHz Pentium III CPUs, 512MB ECC RAM

RAIDZONE CS10R

4U 1.2 TB $15,745

For more information, please visit our Web site:

www.raidzone.com

or give us a call at:

1-800-388-1896

Learn Linux from the experts

We test real skills on live equipment in classes taught by true experts. That's why RHCEs earn an average of 15% more than their non-certified colleagues (CRN Magazine).

And that's why Certification Magazine / Fairfield Research rated RHCE as the top certification (Jan. 2002):

#1 for overall quality of program
#1 for quality of tests and exams
#1 for overall educational quality

Learn locally:

- RH033 Red Hat Linux Fundamentals
- RH133 Red Hat Linux System Administration
- RH253 Red Hat Linux Networking and Security Administration
- RH300 RHCE Rapid Track Course

redhat.
CERTIFIED
ENGINEER

REGISTER NOW FOR RED HAT TRAINING CLASSES IN YOUR AREA:
Call 866-2REDHAT ext. 45733 or visit www.redhat.com/explore/lm

NEW TO LINUX? Newbie to guru—we train at all levels. Call 866-2REDHAT ext. 45733